Modeling Morphogenesis with Reaction-Diffusion Equations using Galerkin Spectral Methods

USNA
Trident Project
Benjamin M. Heineike
Project advisers:
Professor Reza Malek-Madani
Professor Sonia Garcia

Project Abstract

This project studies the nonhomogeneous steady-state solutions of the Gray-Scott model, a system of nonlinear partial differential equations that has received attention in the past decade in the context of pattern formation and morphogenesis. Morphogenesis, or 'birth of shape', is the biological term for the initial formation of patterns that occur in development as cells begin to differentiate. The model is a two morphogen reaction-diffusion system in which individual molecules display complex self-organization in aggregate.

The project is divided into two main parts. The first part develops the Galerkin Spectral method for application to the two species reaction-diffusion system. Limitations and capabilities of the Galerkin Spectral method are discussed in the context of the heat equation, the Burgers equation, and the Allen-Cahn equation.

The second part analyzes the stability of equilibria in the Gray-Scott model in terms of reaction and diffusion parameters. A region of Hopf bifurcation is identified for the diffusionless system, and conditions for diffusion driven instability are developed. We show in particular that diffusion driven instability will occur only when the diffusion constants of each morphogen are different in any two species reaction-diffusion equation. We then show some numerical simulations of pattern formation in the Gray-Scott model using MATLAB programs to implement the Galerkin Spectral method.

Keywords: reaction-diffusion equations, morphogenesis, Gray-Scott model, Galerkin Spectral method, Allen-Cahn equation, the Burgers equation, partial differential equations, numerical simulations, MATLAB.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
06-05-2002		xx-xx-2002 to xx-xx-2002

4. TITLE AND SUBTITLE
Modeling Morphogenesis with Reaction-Diffusion Equations using Galerkin Spectral Methods
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Heineike, Benjamin M. ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
US Naval Academy
Annapolis, MD21402

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT
See report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	89	email from USNA, Annapolis, MD, (blank) lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	6 May 2002	

4. TITLE AND SUBTITLE
Modeling morphogenesis with reaction-diffusion equations using Galerkin spectral methods

5. FUNDING NUMBERS

6. AUTHOR(S)
Heineike, Benjamin M. (Benjamin Murrary), 1980-

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

US Naval Academy
Annapolis, MD 21402

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

Trident Scholar project report no. 296 (2002)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT: This project studies the nonhomogeneous steady-state solutions of the Gray-Scott model, a system of nonlinear partial differential equations that has received attention in the past decade in the context of pattern formation and morphogenesis. Morphogenesis, or 'birth of shape', is the biological term for the initial formation of patterns that occur in development as cells begin to differentiate. The model is a two morphogen reaction-diffusion system in which individual molecules display complex self-organization in aggregate. The project is divided into two main parts. The first part develops the Galerkin Spectral method for application to the two species reaction-diffusion system. Limitations and capabilities of the Galerkin Spectral method are discussed in the context of the heat equation, the Burgers equation, and the Allen-Cahn equation. The second part analyzes the stability of equilibria in the Gray-Scott model in terms of reaction and diffusion parameters. A region of Hopf bifurcation is identified for the diffusionless system, and conditions for diffusion driven instability are developed. We show in particular that diffusion driven instability will occur only when the diffusion constants of each morphogen are different in any two species reaction-diffusion equation. We then show some numerical simulations of pattern formation in the Gray-Scott model using MATLAB programs to implement the Galerkin Spectral method.

14. SUBJECT TERMS
reaction-diffusion equations, morphogenesis, Gray-Scott model, Galerkin Spectral method, Allen-Cahn equation, the Burgers equation, partial differential equations, numerical simulations, MATLAB

15. NUMBER OF PAGES
91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Table of Contents:

Reaction-Diffusion Equations and Morphogenesis using Galerkin Methods

USNA
Trident Project
Benjamin M. Heineike
Project advisers:
Professor Reza Malek-Madani
Associate Professor Sonia Garcia

1. Introduction: Modeling Morphogenesis

In the context of embryology, morphogenesis is, as its Greek roots suggests, the part of the process of development where shape and form begin to emerge. The process by which each nearly identical stem cell makes the initial decision to take the unique pathway that leads it to fulfill its role as a specialized, functional cell in the fully-grown organism is still largely a mystery. After the cells have taken those first steps, their subsequent development is better understood, but the nature of the initial impulse that causes one cell to become a muscle cell and another to become a nerve cell is just beginning to be understood.

Two years before Alan Turing died in 1954, he published a paper entitled *"The Chemical Basis of Morphogenesis"* in the *Philisophical Transactions of the Royal Society of London*[25] that had a much different flavor than the work on computing machines and artificial intelligence for which he was so famous. Today his paper is the cornerstone of a body of scientific literature suggesting that the basis of morphogenesis lies in the pattern formation capabilities of certain chemicals thought to be present in an early embryo, which Turing dubbed morphogens. According to the theory, by the time the genes begin to create proteins in a growing embryo, there is already a chemical pattern of morphogens present in the background of the tissue. These morphogens interact physically and chemically with one another to create patterns. These chemical patterns provide initial developmental signals to the genes that cannot diffuse

themselves between neighboring cells. If the interactions of the morphogens were effected primarily by chemical reactions and physical diffusion, Turing suggested that one could model their pattern formation mathematically using physical and chemical laws. More specifically, one could model morphogenesis with a particular class of partial differential equations called reaction-diffusion equations. In 1991, Lewis Wolpert articulated a theory of positional information in his book *The Triumph of the Embryo*[27] in which genes were influenced by morphogen concentrations distributed in a gradient. In one example the location of each digit on a chick's wing was thought to be pinpointed by a corresponding concentration of the morphogen, retinoic acid, which decreased as it diffused away from a source along the antero-posterior axis of the wing (which is the axis determined by a pencil in one's hand if it were held with a closed fist).

Only recently with the work of researchers like Ouyang, Swinney, and Petrov [17],[19] have experimental "Turing Patterns" been exhibited on actual biological chemicals in a laboratory. The self-organizing patterns that biologists are seeing in their petri dishes and gelatinous solutions have been organizing themselves for years on the computer screens of mathematicians [9],[14],[15],[18]. From the time that Turing first showed that nonhomogeneous solutions existed for reaction-diffusion systems modeling morphogenesis in a simple ring of cells, there has been significant interest in the types of patterns that arise in the solutions of these reaction-diffusion equations.

Reaction-diffusion equations are partial differential equations (PDEs) which model the way collections of particles behave taking into account two forces. First, they model the way that the particles interact with one another at a given point in space, described in chemical terms as their reactions. Second, they model the way that a given particle distributes itself in space,

diffusing from regions of higher concentrations towards regions of lower concentrations. Reaction-diffusion equations can model more than just morphogenesis, having been used in the past to model population densities, host/parasite models [20], electrical reactions that occur between nerve cells [8], and chemical waves such as those found in the Belousov-Zhabotinsky reaction (ch. 7, [15]). Turing's remarkable discovery was the fact that the presence of diffusion, which acts in a region to distribute particles more homogeneously, given the right interactions between particles being modeled, could help initiate very interesting, nonhomogeneous patterns.

Only today with advanced computer technology and mathematical methods are we beginning appreciate the rich pattern formation capabilities of these models. As a general rule, the variability of the initial and boundary conditions as well as the complex interaction between concentrations over both space and time preclude a simple algorithmic solution to a reaction-diffusion system. As with most partial differential equations used in modeling, exact solutions are seldom discovered or even attempted. Therefore, to investigate our reaction-diffusion system, which models two morphogens in two dimensions, we will turn towards approximate solutions. Recent advancements in scientific computing and numerical analysis have increased the availability of high-powered computing resources to researchers, rendering approximate solutions of partial differential equations an efficient and reliable tool.

The particular approximation method that we will use is called the Galerkin Spectral method. Many approximation methods (the Finite Difference method for instance) break the problem into smaller workable bits by dividing the spatial domain into pieces on a mesh and then putting the individual puzzle pieces of the solution back together. Instead of breaking the solutions down into puzzle pieces, spectral methods can be thought of as breaking the solution into layers. We assume that the solution to the system can be written as a linear combination of a

product of unknown functions in time with known basis functions over space. The Galerkin method is a way to determine these unknown functions in time by eliminating spatial dependence using inner products.

Galerkin methods reduce PDEs to a collection of Ordinary Differential Equations (ODEs) in time. For an exact solution, we would need a system of infinitely many ODEs, but for our purposes a finite approximation will be sufficient to capture the overall behavior of the system. The primary advantage of the Galerkin method is that it is relatively easy to program and to increase the number of "layers" in the approximate solution. As will become clear in the subsequent sections, just as one is able to increase the accuracy of a Fourier series representation of a solution by taking more terms in the Fourier basis, we are able to add more accuracy to our approximate solution by taking more terms in the so-called Galerkin basis.

The focus of this project is to create a body of programs in MATLAB that can solve a system of nonlinear reaction-diffusion equations using the Galerkin method, and then use them to analyze the pattern formation capabilities of a particular system in the context of morphogenesis. In particular, we are looking for nonhomogeneous steady-state and periodic solutions of the Gray-Scott model, a particular two species reaction-diffusion system in two dimensions [18]. Along the way we use the method to test certain one-dimensional nonlinear reaction-diffusion equations such as the Burgers equation and the Allen-Cahn equation [4].

For the Gray-Scott model, we will also analyze the ODEs created by the lower degree Galerkin method solutions, and find regions in parameter space near which particular equilibrium points of the solution undergo Hopf bifurcation. We also use the principles of spectral analysis to find conditions on the diffusion parameters that exhibit diffusion driven

instability. This analysis helps us to find parameter values for which the solution exhibits periodic behavior and nonhomogeneous steady-state solutions.

Intrinsic to this study is an awareness of the complexity of these systems. In a mathematical context, complex does not simply mean complicated. Rather, complexity describes the phenomenon whereby complicated structures arise from the simple interactions of interconnected individual particles. The structures that emerge from these interactions are not virtually random like chaotic systems, but are intricately structured. However, the eventual organization of a system will not immediately follow from the rules of interaction for the particles, and could not possibly be extrapolated from an examination of a particular particle by itself. One would use complexity to describe the way that the interaction of each individual ant with its neighbors and the environment gives rise to a seemingly conscious colony, or how individual water and air molecules acting according to physical and chemical principles can form highly ordered weather structures like storm clouds and tornadoes. Our morphogenesis model is yet another application of this concept. We propose that some of the complicated patterns exhibited in living organisms with such a high level of reproducibility are controlled by the simple interactions of morphogens in highly interconnected networks of individual cells. The purpose of studying this type of complexity is to gain clues about the characteristics of an aggregate structure from knowledge of the way individual particles interact, without resorting to isolating the particle from its neighbors. We would like to know not only how an individual particle affects the entire system, but also what characteristics of the entire system cause the particles to organize themselves. First we will define reaction-diffusion equations and the model.

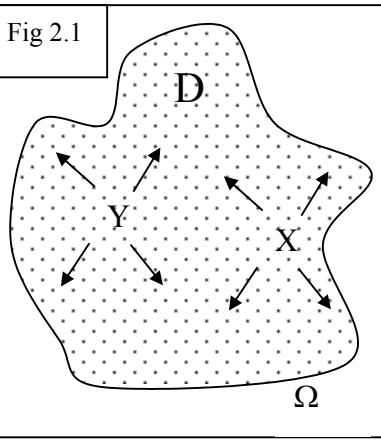
2. Reaction-Diffusion Equations

A system of reaction-diffusion equations is a system of equations of the form

$$\frac{\partial \mathbf{u}}{\partial t} = D\Delta\mathbf{u} + \mathbf{f}(\mathbf{u}, \nabla\mathbf{u}, \mathbf{x}, t) \qquad (2.1)$$

over a region $D \subseteq \mathbf{R}^n$, where $\mathbf{u}(\mathbf{x}, t)$ is a vector representing the states (in our model morphogen concentrations) of a group of substances at time t and position $\mathbf{x} \in \mathbf{R}^n$. A is a matrix of diffusion coefficients, which in a two species system is typically of the form $D = \begin{bmatrix} \mu & 0 \\ 0 & v \end{bmatrix}$, and $\Delta\mathbf{u}$ is the Laplacian differential operator acting on $\mathbf{u}$ with respect to $\mathbf{x} \in D$. It is the second order spatial rate of change of $\mathbf{u}$. In the most general case, the inputs in the reaction function $\mathbf{f}$ are $\mathbf{u}, \nabla\mathbf{u}$, the gradient of $\mathbf{u}$ with respect to $\mathbf{x}$, $\mathbf{x}$ and t (p.5, [1]). These partial differential equations are subject to boundary conditions over $\Omega \subseteq D$ and initial conditions. In these equations the term containing the Laplacian operator is the diffusion term. Without the function $\mathbf{f}$, (2.1) is the heat equation, one of the first equations encountered in any partial differential equation course. The heat equation models the diffusion of heat from regions of higher temperature, or heat concentration, to regions of lower temperature, which is very similar to chemical diffusion. The function $\mathbf{f}$ is called the reaction function because it represents the interactions between particles that act to increase or decrease the quantities of each species, and may depend on the concentration of particles themselves ($\mathbf{u}$), the gradient of the concentrations with respect to space ($\nabla\mathbf{u}$), and the location of the reaction in space and time, ($\mathbf{x}$ and t). The use of chemical terms is meant merely as an analogy, as reaction-diffusion equations

Fig 2.1



have found broad application in areas other than chemistry, such as neurological signal transmission [2], Belousov-Zhabotinsky chemical waves (p159, [2]), geochemical systems (p229, [2]), combustion theory (p114 [2]), and other complex systems. In particular, the Burgers equation,

$$u_t = \varepsilon u_{xx} + uu_x ,$$

can be classified as a reaction-diffusion equation and is one of the most important equations in the study of fluid dynamics. We will use it later to test our numerical methods.

This study is primarily concerned with functions **f** that depend only on the concentrations of the reactants. This idealization is a good approximation for many chemical reactions held at constant temperature, as is often true for biological reactions. The general system now reduces to

$$\frac{\partial \mathbf{u}}{\partial t} = D\Delta \mathbf{u} + \mathbf{f}(\mathbf{u}). \qquad (2.2)$$

This system is augmented by initial conditions,

$$\mathbf{u}(\mathbf{x},0) = \mathbf{h}(x) ,$$

and boundary conditions. We will be dealing with two morphogens, that is, the vector **u** will be given by:

$$\mathbf{u}(\mathbf{x},t) = \begin{bmatrix} u(\mathbf{x},t) \\ v(\mathbf{x},t) \end{bmatrix}.$$

Murray [11] gives a good overview of several reaction-diffusion equations used to model morphogenesis. The three primary reactions he mentions are Schnakenberg's reaction, Gierer and Meinhardt's activator/inhibitor model, and Thomas' experimental model. Schnakenberg's

model is mathematically accessible, but has not found much biological application. It is given in nondimensionalized form by

$$\begin{aligned} u_t &= \gamma(b - v^2 u) + d\Delta u, \\ v_t &= \gamma(a - v + uv^2) + \Delta v. \end{aligned} \tag{2.6}$$

Gierer and Meinhardt [10], [5] proposed the following model which is known as an activator/inhibitor system.

$$\begin{aligned} u_t &= \gamma\left(a - bv + \frac{u^2}{v}\right) + \Delta u, \\ v_t &= \gamma(u^2 - v) + d\Delta v. \end{aligned} \tag{2.7}$$

The nondimensionalized parameters are the same as above. For this equation we will call u the activator, since it acts to increase the population of both chemicals, and v will be the inhibitor, since it decreases the rate of change over time for each morphogen. For patterns to occur, Gierer and Meinhardt showed that $d>>1$, in other words, that the inhibitor must diffuse significantly faster than the activator. As an illustration, consider a predator/prey system [14]. Think of the prey as the activators and the predators as the inhibitors. The predators, cheetahs for instance, 'diffuse' faster than the prey, antelope. Where the antelope gather together they create an environment where more of their kind can thrive, but the fast moving cheetahs inhibit their numbers (through digestion) when they stray from the herd. Also contact between antelope and cheetahs (again through digestion) activates the production of cheetahs. For the right parameters, activator/inhibitor reaction-diffusion systems form dappled patterns where activators clump together that can be thought of as analogous to herds of prey species.

The reaction-diffusion system that we will focus on is called the Gray-Scott Model. It has been studied numerically by Pearson, who used a finite difference approach [18]. The system is given by

$$\begin{aligned} u_t &= d_u \Delta u - uv^2 + F(1-u), \\ v_t &= d_v \Delta v + uv^2 - (F+k)v, \end{aligned} \tag{2.8}$$

where k is the dimensionless rate constant, F is the dimensionless feed rate, and d_u and d_v are the diffusion coefficients. For our simulations they will typically be d_u=2 x 10^{-5} and d_v=10^{-5} which are the same values that Pearson used. Our goal is to understand the effect of the parameters F, k, d_u and d_v on the long term behavior of the solutions of the initial-boundary value problem associated with (2.8). We would also like to understand the effects these parameters have on the accuracy of our numerical approximation method. More concisely, we would like to know what causes patterns to form in (2.8) and in reaction-diffusion equations in general.

3. The Galerkin Spectral Method

Our method of numerically simulating reaction-diffusion equations is the Galerkin Spectral method. Whereas many of the most common PDE approximation methods used today, including the finite difference method, discretize the problem in space, the Galerkin and other spectral methods discretize the problem over a *spectrum* of functions that are continuous over the whole space. In that sense spectral methods are more global in nature. This spectrum of functions, which is chosen according to the boundary conditions, forms an orthogonal basis for the function space in which we seek the solution of the PDE. To implement spectral methods we construct an approximate solution that is a linear combination of unknown time functions coupled with known spatial basis functions that satisfy the boundary conditions associated with the PDE. The procedure is analogous to using Fourier analysis to express any bounded continuous function as a linear combination of sine and cosine functions at various frequencies.

The concept of finding a set of basis functions with which we can represent all manner of functions that satisfy the boundary conditions is central to spectral methods, and should be explained further. In finite dimensional vector spaces, a basis is a linearly independent set of vectors such that any vector in the vector space can be written as a linear combination of the basis vectors. To illustrate, let V be a finite dimensional vector space over the field A and $B=(\beta_1, \beta_{2,\ldots,} \beta_m)$ be a basis in V. Then for each v in V there are constants $a_i \in A$, also called the coordinates of v in the basis, such that

$$v = a_1\beta_1 + a_2\beta_2 + \ldots + a_m\beta_m . \qquad (3.1)$$

Since (3.1) is true for all vectors v in V then we say that B spans the vector space. For B to form a basis of V, it must do more than just span the entire vector space. B must also be a be linearly

independent set, meaning that no single basis vector in the set can be written in terms of the others. In that sense, a basis is the smallest sized set that can span a vector space. We often choose basis vectors in such a way that they are mutually orthogonal, that is if (a,b) represents the inner product then

$$(\beta_i, \beta_j) = 0 \text{ when } i \neq j. \qquad (3.2)$$

Equation (3.2) will help us compute the coordinates a_i from (3.1). Taking the inner product of both sides of (3.1) with a fixed basis vector β_i yields $(v, \beta_i) = a_i(\beta_i, \beta_i)$ or

$$a_i = \frac{(v, \beta_i)}{(\beta_i, \beta_i)}. \qquad (3.3)$$

An important property of a set of basis vectors is that it helps define what is called a weak characterization of the zero vector:

$$v \equiv 0 \quad \text{if and only if } (v, \beta_i) = 0 \text{ for all } i. \qquad (3.4)$$

We will see later that a similar characterization for infinite dimensional function spaces will prove essential to the Galerkin Spectral method. For more on vector spaces and basis sets, see Chapter 2 of [7].

Finite dimensional vector spaces differ from their infinite-dimensional counterparts in one important respect. In finite dimensions, as long as the number of linearly independent vectors in a set is equal to the dimension of the vector space, the set will span the entire space. The fact the vectors in the set are linearly independent and span the vector space means that the set is a basis for the vector space. In an infinite dimensional vector space, however, we cannot be sure that a particular set is a basis even if it contains infinitely many linearly independent vectors because we do not know whether or not the vectors span the space. The space in which

the solutions to our partial differential equations lie, the set of bounded functions defined over the domain Ω which satisfy the boundary conditions, is an infinite dimensional function space. We must, therefore, be thorough when determining whether or not a proposed set of basis functions spans a vector space.

Before we go any further, let us define orthogonality for our infinite dimensional function space. Let $f(x)$ and $g(x)$ be real-valued functions in the space. Their inner product (f,g) is defined as

$$(f,g) = \int_{\Omega} f(x)g(x)dx . \qquad (3.5)$$

When f and g are orthogonal this inner product is zero.

Let $C = \{\phi_1, \phi_2, \phi_3, ...\}$ be an orthogonal set of basis functions from the function space G. Since the functions in C are orthogonal, they automatically satisfy the linear independence condition. In addition the set C must also be complete in G, that is, it must also be possible for any function f in G to be expressed as a linear combination of this infinite set of basis functions. More concisely, for each f in G,

$$f(x) = \sum_{i=1}^{\infty} a_i \phi_i(x) \qquad (3.6)$$

for some constants $\{a_1, a_2, a_3, \ldots\}$.

Equality in (3.6) means:

$$\lim_{M\to\infty} \left\| f(x) - \sum_{i=1}^{M} a_i \phi_i(x) \right\| = 0 ,$$

where

$$\|g\|^2 = (g,g) = \int_{\Omega} |g(x)|^2 dx .$$

Let H be the set of all one-dimensional, square integrable, real-valued functions defined on [0,1] that vanish at the endpoints:

$$H = \{f : R \to [0,1] \mid \ \|g\| < \infty,\ f(0) = f(1) = 0\}.$$

A well-known result from Fourier analysis states that the set

$$C = \{\sin(\pi x), \sin(2\pi x), ..., \sin(n\pi x), ...\} \qquad (3.7)$$

forms a basis for H ([6] sec. 5.3.3). Just as we expressed any vector in a vector space in terms of coordinates with respect to a set of basis vectors, given by (3.3), we can express any function in H in terms of coordinates with respect to C. Using the orthogonality of $\sin(i\pi x)$ over the domain [0,1], which means

$(\sin(i\pi x), \sin(j\pi x))=0$ if $i \neq j$, (3.8)

we see that

$$a_i = \frac{(f(x), \phi_i(x))}{(\phi_i(x), \phi_i(x))}. \qquad (3.9)$$

Notice that if we remove any of the functions in C with an even coefficent inside the sine function to form a new set C^*, we will still have infinitely many orthogonal functions that satisfy the boundary conditions, but will no longer have a basis for H since not every function in H can be written as a linear combination of the functions in C^*. For example, consider removing $\sin(2\pi x)$ from C to generate C^*. If C^* were a basis for H then we could write $\sin(2\pi x)$ in terms of its coordinates. However, each of the a_i are determined by (3.9):

$$a_i = \frac{(\sin(2\pi x), \sin(i\pi x))}{(\sin(i\pi x), \sin(i\pi x))} = 0,$$

where i can be every natural number except for 2. Since each of the coordinates is 0, then $\sin(2\pi x)$ would have to be the 0 function, a contradiction. C^* cannot, therefore, be a basis for H even though it consists of infinitely many linearly independent functions in H.

The Galerkin Spectral method searches for solutions of the PDE system in terms of linear combinations of the basis functions multiplied by unknown functions in time. The fact that the domain on which we are looking for the solutions is bounded assures us that we will be able to form a basis with countably many basis functions. The unknown functions in time are then found by solving the now countably many ODEs that are created by substituting the template solution into the PDE.

For the reaction-diffusion system given by (2.2)

$$\frac{\partial u}{\partial t} = D\Delta u + f(u) \tag{3.10}$$

we define the differential operator

$$L(u) = u_t - D\Delta u - f(u). \tag{3.11}$$

Note that (3.10) is equivalent to

$$L[u]=0. \tag{3.12}$$

We search for solutions to (3.10) of the form:

$$\tilde{u}(x,t) = \sum_{i=1}^{\infty} a_i(t)\phi_i(x), \tag{3.13}$$

where the functions $\phi_i(x)$ satisfy the appropriate boundary conditions. To find the unknown $a_i(t)$, we now substitute (3.13) into (3.12) and use the characterization of zero similar to (3.4):

$$g(x) \equiv 0 \text{ if and only if } (g(x), \phi_i(x)) = 0 \text{ for all } i. \tag{3.14}$$

This characterization is computationally useful because in order to show that a function is zero for each value in its continuous domain, one must only show that the countably many (but infinite) conditions in (3.14) are satisfied, namely that the inner product of each basis function with g is zero. Setting $L(\tilde{u}) = 0$ in this sense, we are left with the countable conditions:

$$\begin{aligned} &(\phi_1, L[\tilde{u}]) = 0, \\ &(\phi_2, L[\tilde{u}]) = 0, \\ &\vdots \end{aligned} \qquad (3.15)$$

The inner product operation in (3.15) removes spatial dependence and these conditions leave us with a system of ODEs in terms of the unknown coefficients $a_i(t)$. In practice, instead of using infinitely many basis functions in (3.13), we truncate the solution template at some finite number N. Thus in (3.15) we would have N ODEs in N unkowns. To find the initial conditions needed to solve (3.15), we use the initial conditions in the original PDE,

$$u(x,0)=h(x) .$$

Since (3.13) must satisfy these initial conditions as well, we impose the following conditions on the $a_i(t)$:

$$\sum_{i=1}^{\infty} a_i(0)\phi_i(x) = \tilde{u}(x,0) = u(x,0) = h(x).$$

Thus the initial condition for each function $a_i(t)$, $a_i(0)$, is given by its coordinate for $h(x)$ in terms of the basis functions. By (3.9) we have an explicit formula;

$$a_i(0) = \frac{(h(x), \phi_i(x))}{(\phi_i(x), \phi_i(x))} .$$

Now that we have N ODEs in N unknowns with initial conditions, we can solve for the $a_i(t)$ and then use them to construct an approximate solution for the PDE. We can solve these

ODEs numerically using any number of ODE solving packages found in mathematical software such as MATLAB and Mathematica.

We will now illustrate the steps associated with the implementation of the Galerkin method on the following basic reaction-diffusion equation in one space dimension:

$$u_t = du_{xx} + f(u), \tag{3.16}$$

where d is a constant real number. The spatial domain is D=[0,1]. The equation in (3.16) is subjected to Dirichlet boundary conditions,

$$u(0,t) = u(1,t) = 0, \tag{3.17}$$

and initial conditions,

$$u(x,0) = g(x). \tag{3.18}$$

Step 1. Choose the basis functions and solution template: The primary precondition for this choice is that the basis functions satisfy the boundary conditions. If $C = \{\phi_1, \phi_2, \phi_3, ...\}$ is our basis then the solution template is of the form:

$$\tilde{u}(t,x) = \sum_{n=1}^{N} a_n(t)\phi_n(x). \tag{3.19}$$

For the boundary conditions in this example we will choose

$$\phi_n(x) = \sin(n\pi x), \tag{3.20}$$

where n is an integer since (3.20) is zero when x=1 and x=0. Other considerations for this choice have to do with the type of solution that one expects, the information about the solution that we are trying to extract, and of course the level of complexity that our numerical simulator is prepared to handle. Certain basis functions may be more natural for the solution; a continuous solution would probably be better represented in a continuous basis set of functions. The individual modes of the basis functions may have physical interpretations for the model, and we can choose a basis set that takes advantage of these interpretations to reveal certain properties about the model. Later, when we look at the Gray-Scott model, the 0 mode solution will represent the system without diffusion, and we can 'add' diffusion to our approximations by taking the solution with more modes. Table (3.1) gives examples of Fourier basis functions one can use for the boundary conditions listed in the left column.

Table 3.1

Type of Boundary Conditions	**Fourier basis to use**
Periodic	$e^{in\pi x}$
Dirichlet: u(0,t)=0, u(1,t)=0	$\sin(n\pi x)$
Neumann: u_x(0,t)=0, u_x(1,t)=0	$\cos(n\pi x)$
Mixed: u_x(0,t)=0, u(1,t)=0	$\cos\left(\frac{(1+2n)\pi x}{2}\right)$

There are a variety of basis functions that will work with a given set of boundary conditions. If we did not want to use Fourier basis functions for one reason or another, we could look for Chebyshev and Legendre polynomial bases as well as wavelets.

Step 2. Substitute template into differential operator and obtain a set of ODEs in time: Our differential operator will be

$$L(u) = u_t - du_{xx} - f(u). \qquad (3.21)$$

Substituting $\tilde{u}$ from (3.19) into (3.21) we have:

$$L(\tilde{u}) = \sum_{n=1}^{N} a_n'(t)\sin(n\pi x) + d\left(\sum_{n=1}^{N} n^2\pi^2 a_n(t)\sin(n\pi x)\right) - f\left(\sum_{n=1}^{N} a_n(t)\sin(n\pi x)\right). \qquad (3.22)$$

Notice that the second derivative has been replaced in (3.22) with the constant $-n^2\pi^2$ because $\left(\sin(n\pi x)\right)_{xx} = -n^2\pi^2\sin(n\pi x)$. In order to obtain the $a_n(t)$, we set (3.22) equal to zero using the weak formulation described in (3.14) and (3.15):

$$\sum_{n=1}^{N} a_n'(t)\left(\sin(m\pi x),\sin(n\pi x)\right) + \pi^2 d\sum_{n=1}^{N} a_n(t)n^2\left(\sin(m\pi x),\sin(n\pi x)\right) - \left(\sin(m\pi x), f\left(\sum_{n=1}^{N} a_n(t)\sin(n\pi x)\right)\right) = 0$$

$$m = \{1,2,...,N\}. \qquad (3.23)$$

Notice that the inner product operation and the summation operations commute because integration is a linear operation. The functions $\phi_n(x) = \sin(n\pi x)$ are orthogonal on the interval [0,1] and when $m=n$, $(\sin(m\pi x), \sin(n\pi x))=1/2$ so (3.23) now takes the form:

$$\frac{a_m'(t)}{2} = \frac{-\pi^2 m^2 d}{2} a_m(t) - \left(\phi(x)_m, f\left(\sum_{n=1}^{N} a_n(t)\phi_n(x)\right)\right) \qquad (3.24)$$

for m=1,...,N.

Step 3. Find the Initial conditions for the ODEs: In order to find a unique solution to a system of ordinary differential equations, we need initial conditions for each function in the system. For the initial conditions, we use (3.9) to get

$$a_m(0) = \frac{\left(\sin(m\pi x), g(x)\right)}{\left(\sin(m\pi x), \sin(m\pi x)\right)} = 2\left(\sin(m\pi x), g(x)\right) \qquad (3.25)$$

for m=1,...,N.

Step 4. Solve the initial value problem for time: Now we have N ODEs in N unknowns with initial conditions. If the function f is nonlinear, then the system of ODEs will be nonlinear, and we will be forced in most cases to look for an approximate solution using a numerical ODE solver such as MATLAB's ODE45 or Mathematica's NDSolve. The main computational costs will arise from the last term in (3.24) which is the nonlinear term. We use forward time stepping methods primarily, such as the Runge-Kutta methods built into the MATLAB and Mathematica ODE solvers. Many reaction-diffusion equations have a property called stiffness that causes forward time stepping methods to become unstable unless the time step is made extremely small. MATLAB and Mathematica use highly adaptive routines that can usually avoid this problem; however, the computational cost of solving stiff differential equations could become prohibitive even for adaptive solvers.

Step 5. Reconstruct the solution: Once we have computed all the coordinate functions $a_n(t)$, we can reconstruct the approximate solution using (3.19). Since we solved the ODEs computationally, our $a_n(t)$ will not be continuous functions but rather discretized tables of $a_n(t)$ values given at time intervals specified by our numerical ODE solving scheme. Errors in our

final approximate solution will arise at several points during the process. First and foremost, they will be due to choosing N too small. The magnitude of N that one must use to bound the error at a given value for any particular reaction-diffusion equation is a difficult problem in analysis that will not be addressed here. Errors will also arise from numerically approximating solutions to the ODEs. When comparing the approximate solution to actual reaction-diffusion systems found in nature, we must also take into account errors that arise from approximations and simplifications that were made when creating the mathematical model.

The tacit assumption made throughout this study is that our initial-boundary value problem has a unique solution. This assumption can be made rigorous in the setting of some of our problems (for example, the nonlinear Allen-Cahn equation) but remains a formidable open problem in mathematical analysis for others including the Burgers equation and the Gray-Scott model. Indeed the complex structure of the periodic and steady-state numerical solutions we obtain is an indication of how difficult it will be to develop an existence and uniqueness theory for nonlinear PDEs.

Example 1: One-dimensional heat equation

The simplest reaction-diffusion equation is one that does not contain any reaction at all. This equation is commonly referred to as the heat equation since it has been used extensively to model the way heat spreads in various media over time. It is given in one dimension by (3.16) where $f(u)=0$:

$$u_t = ku_{xx}, \qquad x \in (0, L) \qquad (3.26)$$

which can be expressed in operator notation as:

$$L(u) = u_t - ku_{xx} = 0 \qquad (3.27)$$

with initial conditions

$$u(x,0)=g(x). \qquad (3.28)$$

In (3.26) k represents the thermal diffusivity constant. We consider here Dirichlet boundary conditions

$$u(0,t) = u(L,t) = 0 \qquad (3.29)$$

and follow the step by step process outlined above.

Step 1- Choose the basis functions and solution template: Since we have Dirichlet boundary conditions, we choose $\phi_n = \sin(\frac{n\pi x}{L})$. Our solution template is:

$$\tilde{u}(t,x) = \sum_{n=1}^{N} a_n(t)\sin(\frac{n\pi}{L}x). \qquad (3.30)$$

Step 2- Substitute template into differential operator and obtain a set of ODEs in time: Substituting (3.30) into (3.27) we have

$$\sum_{n=1}^{N} a_n'(t)\sin(\frac{n\pi}{L}x) + k\left(\sum_{n=1}^{N}\left(\frac{n\pi}{L}\right)^2 a_n(t)\sin(\frac{n\pi}{L}x)\right) = 0.$$

Using the weak characterization of 0 given in (3.14) and (3.15), this expression gives us the N conditions:

$$\sum_{n=1}^{N} a_n'(t)\left(\sin(\frac{m\pi}{L}x),\sin(\frac{n\pi}{L}x)\right) + k\sum_{n=1}^{N} a_n(t)\left(\frac{n\pi}{L}\right)^2\left(\sin(\frac{m\pi}{L}x),\sin(\frac{n\pi}{L}x)\right) = 0\,. \qquad (3.31)$$

Using the fact that:

$$\left(\sin\left(\frac{m\pi}{L}x\right),\sin\left(\frac{n\pi}{L}x\right)\right)=\begin{cases}L/2 & m=n\\ 0 & m\neq n\end{cases} \qquad (3.32)$$

(3.31) is reduced to

$$a_m'(t)=-k\left(\frac{m\pi}{L}\right)^2 a_m(t) \qquad m=1,\ldots,N \qquad (3.33)$$

since the constant $L/2$ factors out of either side.

Step 3- Find the Initial conditions of the ODEs: Using (3.9) and the orthogonality of our basis set, we have:

$$a_m(0)=\frac{\left(\sin(\frac{m\pi}{L}x),g(x)\right)}{\left(\sin(\frac{m\pi}{L}x),\sin(\frac{m\pi}{L}x)\right)}=\frac{2}{L}\int_0^L \sin(\frac{m\pi}{L}x)g(x)dx. \qquad (3.34)$$

The $a_m(0)$ are nothing more than the Fourier sine coefficients of *g(x)*.

Step 4- Solve the initial value problem for time:

Each of the ODEs in (3.33) has the solution

$$a_m(t)=a_m(0)e^{-k\left(m\pi/L\right)^2 t}$$

which can be verified by direct substitution.

Step 5- Reconstruct the solution: Using our solution template, (3.30), we have

$$\tilde{u}(t,x) = \sum_{n=1}^{N} a_n(0) e^{-k\left(n\pi/L\right)^2 t} \sin(\frac{n\pi}{L}x),$$

where $a_n(0)$ is the nth Fourier sine coefficient of the initial condition. This is indeed the N-th partial sum of the true solution of the heat equation with Dirichlet boundary conditions which can be derived by more traditional methods such as separation of variables.

Example 2: The Burgers equation - *A one-dimensional nonlinear reaction-diffusion equation*

We will now demonstrate how the implementation changes when a nonlinear reaction is added to the problem. A common nonlinear reaction-diffusion equation used in the modeling of turbulence and airflow is the Burgers equation,

$$L(u) = u_t - \varepsilon u_{xx} + uu_x = 0 . \qquad (3.35)$$

In the Burgers equation the quantity u is related to the density of the fluid being modeled and the term εu_{xx} represents viscous dissipation in the model. It is known (see Malek-Madani [13], 418-427) that when ε = 0 that the typical initial disturbances in u form discontinuities known as shock waves and yet when ε>0 solutions that start out smooth remain smooth for all time. The computational challenge in this problem is to gain insight into the Galerkin method's response for small but positive viscosities. We will analyze (3.35) over the interval [0,1] subject to Dirichlet boundary conditions on the same domain as the heat equation above (3.29) with L=1. The initial condition we consider is:

$$u(x,0) = g(x) = \begin{cases} 1-\cos(8\pi x) & x \in [1/4, 1/2] \\ 0 & \textit{otherwise.} \end{cases} \qquad (3.36)$$

Just as before, the boundary conditions lead us to choose the basis $\phi_j = \sin(j\pi x)$. Proceeding through the Galerkin method using steps 1-5 and the relationship (3.32), we arrive at the equations:

$$\frac{a_m'(t)}{2} = \frac{-\pi^2 m^2 \varepsilon}{2} a_m(t) - \left(\phi_m(x), \sum_{n=1}^{N} a_n(t)\phi_n(x) \sum_{j=1}^{N} a_j(t)\phi_j'(x) \right)$$

We can simplify the nonlinear term for ease of programming:

$$\left(\phi_m(x), \sum_{n=1}^{N} a_n(t)\phi_n(x) \sum_{j=1}^{N} a_j'(t)\phi_j'(x) \right) = \sum_{n=1}^{N}\sum_{j=1}^{N} a_n(t)a_j'(t)\left(\phi_m(x), \phi_n'(x)\phi_j(x)\right) = aP_m a'$$

where a is a 1xN row vector whose entries are the a_i, i=1,..,N, and P is the matrix defined by

$$P_m(i,j) = \left(\phi_m(x), \phi_i'(x)\phi_j(x)\right)$$

It is easy to show that

$$P_m(i,j) = \begin{cases} -i\pi/4, & j+m=i, \\ i\pi/4, & |j-m|=i, \\ 0, & (j+m \neq i) \text{ and } (|j-m| \neq i). \end{cases}$$

Therefore, the Galerkin method reduces to N differential equations in N unknowns of the form

$$a_m'(t) = -\pi^2 m^2 \varepsilon a_m(t) - 2(aP_m a').$$

The initial conditions for the above system are given by (3.9) as

$$a_m(0) = \frac{\left(\sin(m\pi x), g(x)\right)}{\left(\sin(m\pi x), \sin(m\pi x)\right)} = 2\int_{1/4}^{1/2} \sin(m\pi x)(1-\cos(8\pi x)dx.$$

One of the main questions we have regarding the Galerkin Spectral method has to do with the magnitude of N that we must pick to accurately describe the system. This choice will depend on many parameters, the most important being the amount of information we need to extract from

our approximation. Figure 3.2 shows Galerkin approximations to the Burgers equation for 8, 32 and 64 basis functions at time 1. We see that as *N* is increased, the approximate solution becomes smoother and displays less oscillations due to the basis functions.

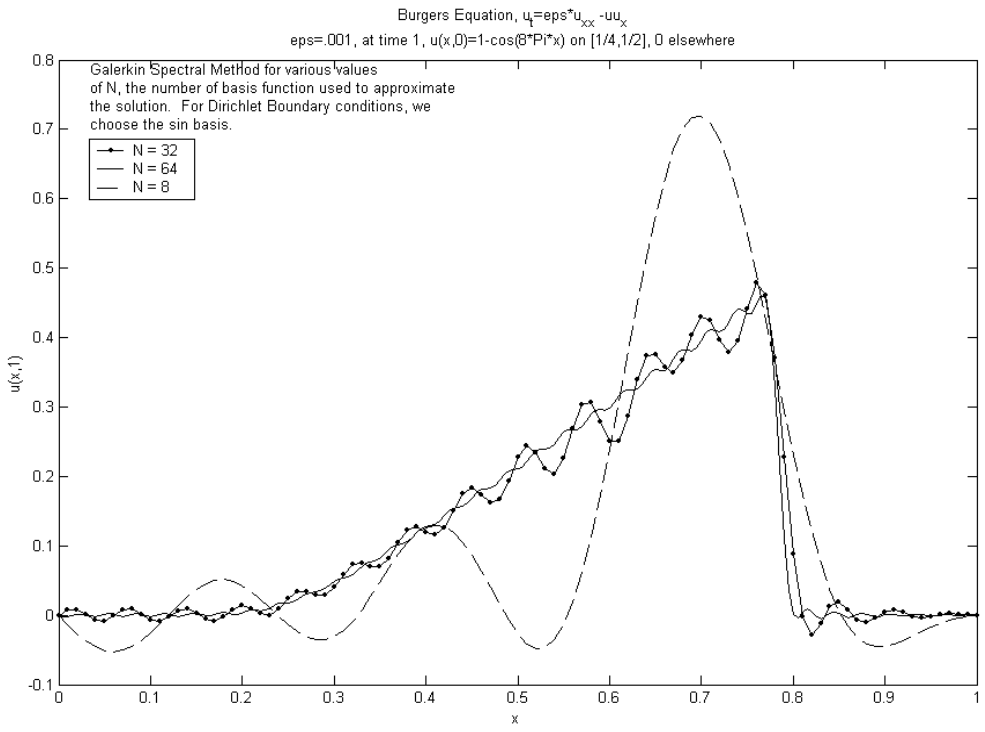


Figure 3.2 Galerkin approximation to the Burgers equation for various N, ε=.001, t=1.

Figures 3.3-3.5 show Galerkin Spectral method approximations to the Burgers equation for various *N* and ε values. In these diagrams, each curve represents our approximate solution at time intervals of 0.05 units with a final time of 1 unit. With *N*=32 and ε=0.01, the solution seems to qualitatively follow the behavior that we would expect from the Burgers equation (Fig 3.3). There are only slight oscillations near the base of the cosine function in the initial condition for the first time value due to the inability of our smooth basis functions to accurately approximate the sharp corners of the non-smooth initial condition. In Figure 3.4 where we decrease ε to 0.001, and keep *N* at 32, the Galerkin method shows much more oscillations and

inaccuracies than it did with $\varepsilon = 0.01$ in Fig 3.3. However, when we increase N to 96, we see that the solution begins to behave more smoothly despite some lingering instabilities. When ε is increased, the solution at time one has a sharper corner and the peak of the solution seems to decrease at a slower rate. This is because ε is inversely proportional to the Reynold's number.

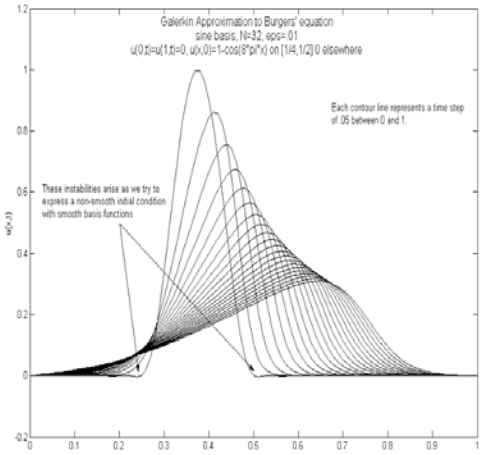


Figure 3.3 The Burgers equation n=32, ε=.01

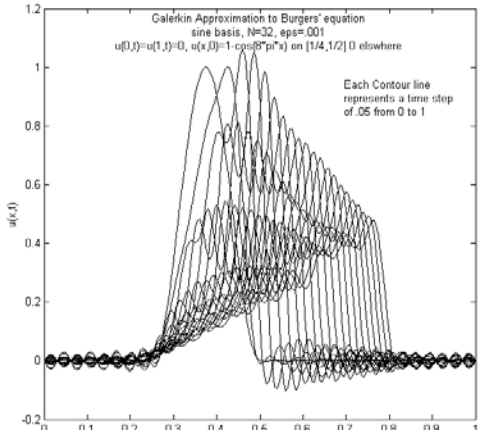


Figure 3.4 Burgers' equation n=32, ε=.001

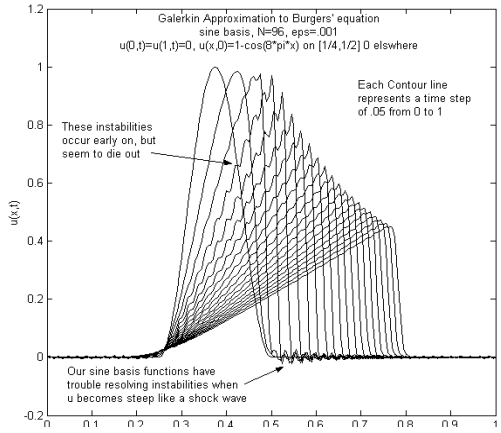


Figure 3.5 Burgers' equation n=96, ε=.001

An ε value of zero would model a fluid with no viscosity (inviscid flow), and a shock wave would form. With a positive ε value, the shock dissipates. Notice also that the oscillations, which are caused by the inability of our basis functions to adapt to non-smooth shock wave behavior begins to form for smaller ε values decreases with time as a result of the diffusion term. As ε is decreased, the effect of the diffusion term is decreased, so we expect oscillations to occur earlier and last longer. This evidence clearly indicates that our choice of N depends on the parameter ε in the equation.

Example 3: The Allen-Cahn Equation

This next example illustrates the use of nonhomogeneous boundary conditions as well as the difficulty of knowing when one has reached a steady-state solution for some reaction-diffusion systems. The Allen-Cahn Equation is given by

$$L(u) = u_t - \varepsilon u_{xx} - u(1 - u^2) = 0. \qquad (3.37)$$

Referring to (3.10) we have $f(u)=u(1-u^2)$. Just as Trefethen does in [23], we will consider the nonhomogeneous Dirichlet boundary conditions:

$$u(-1,t) = -1, \quad u(1,t) = 1, \qquad (3.38)$$

and the initial condition

$$u(x,0) = g(x) = 0.53x + 0.47 \sin(-\frac{3}{2}\pi x). \qquad (3.39)$$

Note that the initial condition satisfies the boundary conditions. Additionally, $g(x)$ is asymmetric about the origin.

Step 1- Choose the basis functions and solution template:

With nonhomogeneous Dirichlet boundary conditions, we choose the solution template in the following form:

$$\tilde{u}(t,x) = x + \sum_{n=1}^{N} a_n(t)\sin(n\pi x). \qquad (3.40)$$

Adding the term x to the solution template ensures that it satisfies the boundary conditions (3.38). This choice is by no means unique, so we have added x, the first function that came to mind that satisfied the conditions. The basis functions will still be $\phi_n(x) = \sin(n\pi x)$ but notice that the solution template is not made up entirely of linear combinations of the basis functions.

Step 2- Substitute the template into the differential operator and obtain a set of ODEs in time:

Performing the procedures used in (3.22) and (3.23) we are left with ODEs of the form:

$$\sum_{n=1}^{N} a_n'(t)\left(\phi_m(x),\phi_n(x)\right)+\pi^2 d\sum_{n=1}^{N} a_n(t)n^2\left(\phi_m(x),\phi_n(x)\right)+\left(\phi_m(x),f\left(x+\sum_{n=1}^{N} a_n(t)\phi_n(x)\right)\right)=0$$

$m=1,\ldots,N.$ (3.41)

Since the basis functions are orthogonal, (3.41) reduces to:

$$a_m'(t)=-\pi^2 dm^2 a_m(t)+\left(\phi_m(x),f\left(x+\sum_{n=1}^{N} a_n(t)\phi_n(x)\right)\right). \qquad (3.42)$$

The last term in (3.42) is the most complicated term and will be the most costly to evaluate numerically since *f* is a nonlinear function.

Step 3- Find the Initial conditions of the ODEs: Paralleling the procedure we used to get (3.9), we must now take into consideration the function *x* that we added to satisfy the boundary conditions. We start with the initial condition,

$$u(x,0)=g(x)=x+\sum_{n=1}^{N} a_n(0)\sin(n\pi x),$$

and take the inner product with each basis function:

$$\sum_{n=1}^{N} a_n(0)\left(\sin(m\pi x),\sin(n\pi x)\right)=\left(\sin(m\pi x),g(x)-x\right).$$

Since the basis functions are orthonormal, meaning that they are orthogonal and $(\phi_n,\phi_n)=1$ for all *n*, our initial conditions are:

$$a_m(0)=\left(\sin(m\pi x),[g(x)-x]\right). \qquad (3.43)$$

Steps 4 and 5 - Solve the initial value problem and reconstruct the solution

Now as above, we can solve the initial value problem (3.42)-(3.43) in MATLAB (see appendix).

Figures 3.6 and 3.7 show the approximate solution using the Galerkin method with 8 basis functions. These figures exhibit an important phenomenon known as bistability found in some

reaction-diffusion systems. Notice in Figure 3.7 that before time 45, the solution seems to have settled into a nonhomogeneous, wave-like steady-state solution, but suddenly snaps into its final nonhomogeneous steady-state solution where u transitions once from –1 to 1. This property of reaction-diffusion equations makes it difficult to say using only computational evidence when a solution has reached its steady state and when it is merely lingering in some intermediate state.

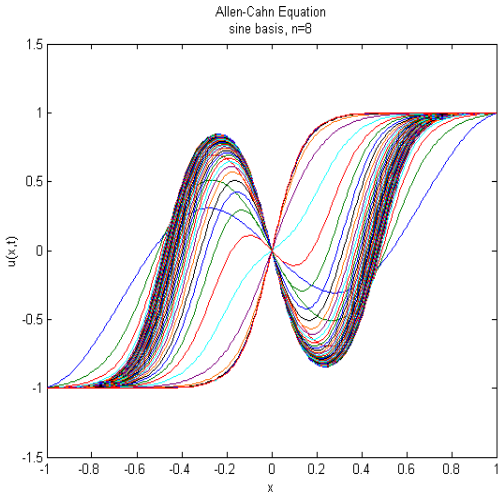


Figure 3.6 Allen-Cahn Equation

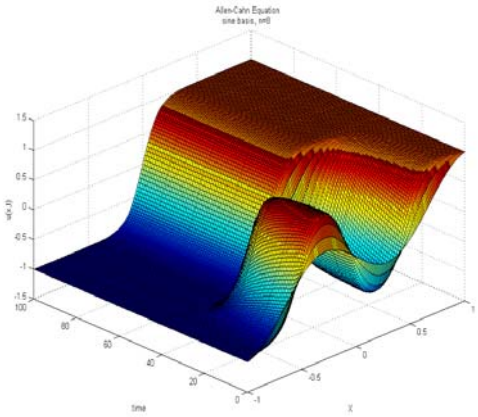


Figure 3.7 Allen-Cahn Equation

Example 4: The heat equation in two dimensions

A few changes are required to reformulate our problem for two dimensions. We consider the heat equation in two dimensions with the heat diffusivity constant, k in (3.27) equal to one:

$$L(u(x,y,t)) = u_t - \Delta u = 0 \qquad (x,y) \in \Omega = [-1,1] \times [-1,1], t \in \mathbf{R}^+ \qquad (3.44)$$

with initial conditions:

$$u(x,y,0)=g(x,y) \qquad (3.45)$$

and periodic boundary conditions. The solution template takes the form:

$$\tilde{u}(x,y,t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y)\,. \qquad (3.46)$$

As in one dimension, the basis functions $\phi_{mn}(x,y)$ are determined by the boundary conditions. To satisfy periodic boundary conditions we choose:

$$\phi_{mn}(x,y) = e^{i(m\pi x + n\pi y)}. \qquad (3.47)$$

Notice that

$$\phi_{mn}(x,y) = \left(i\sin(m\pi x) + \cos(m\pi x)\right)\left(i\sin(n\pi y) + \cos(n\pi y)\right)$$

and

$$\Delta\phi_{mn}(x,y) = -m^2\pi^2 e^{i(m\pi x + n\pi y)} - n^2\pi^2 e^{i(m\pi x + n\pi y)} = -(m^2\pi^2 + n^2\pi^2)\phi_{mn}(x,y). \qquad (3.48)$$

In other words, ϕ_{mn} is an eigenfunction of the Laplacian operator with eigenvalues $-(m^2 + n^2)\pi^2$.

Substituting (3.46) into (3.44) and using (3.48), we have:

$$L(\tilde{u}) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a'_{mn}(t)\phi_{mn}(x,y) + \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\pi^2(m^2 + n^2)\phi_{mn}(x,y). \qquad (3.49)$$

We now take the inner product of (3.49) over Ω with ϕ_{mn}. In two dimensions, where complex valued functions are allowed, the inner product is defined by

$$(f(x,y), g(x,y)) = \iint_{\Omega} f(x,y)\overline{g(x,y)}\,dA,$$

where $\overline{g(x,y)}$ represents the complex conjugate of the function $g(x,y)$. Approximating the infinite sums with finite sums between $-N$ and N, we will be left with $(2N+1)^2$ ordinary differential equations in t of the form:

$$\left(\sum_{m=-N}^{N} \sum_{n=-N}^{N} a'_{mn}(t)\phi_{mn}(x,y), \phi_{pq}(x,y) \right) = \left(\sum_{m=-N}^{N} \sum_{n=-N}^{N} a_{mn}(t)\pi^2(-m^2-n^2)\phi_{mn}(x,y), \phi_{pq}(x,y) \right)$$

or equivalently:

$$\sum_{m=-N}^{N} \sum_{n=-N}^{N} a'_{mn}(t)\left(\phi_{mn}(x,y), \phi_{pq}(x,y)\right) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} a_{mn}(t)\pi^2(-m^2-n^2)\left(\phi_{mn}(x,y), \phi_{pq}(x,y)\right).$$

(3.50)

We see that the inner product,

$$\left(\phi_{mn}(x,y), \phi_{pq}(x,y)\right),$$

is a known quantity for any given *p,q,m,* and *n* and thus we have a system of $(2N+1)^2$ ODEs in $(2N+1)^2$ unknowns (the $a_{mn}(t)$). For our region Ω, the following relationship holds:

$$\left(\phi_{mn}(x,y), \phi_{pq}(x,y)\right) = \int_{-1}^{1}\int_{-1}^{1} e^{i(m\pi x+n\pi y)}\overline{e^{i(p\pi x+q\pi y)}}dxdy = \begin{cases} 4, & (m=p)\,and\,(n=q), \\ 0, & (m\neq p)\,or\,(n\neq q). \end{cases} \qquad (3.51)$$

Using (3.51) we can rewrite (3.50) as

$$a'_{mn}(t) = a_{mn}(t)\pi^2(-m^2-n^2), \qquad (3.52)$$

$m, n \in \{-N, ..., N\}$. To find the initial conditions with orthogonal basis functions we use the same process as in the one-dimensional case. Evaluating (3.46) at t=0, and equating the initial condition of the solution template to that of the true solution given by (3.45), we have:

$$\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(0)\phi_{mn}(x,y) = \tilde{u}(x,y,0) = u(x,y,0) = g(x,y). \quad (3.53)$$

We take the inner product of (3.53) with each basis function, and for each ϕ_{ij} every one of the $a_{pq}(0)$ disappears except $a_{ij}(0)$, resulting in

$$a_{ij}(0) = \frac{1}{4}(g(x,y), \phi_{ij}(x,y)). \quad (3.54)$$

With a system of ODEs and initial conditions, we again have an initial value problem for which we can obtain a numerical solution. This system is much larger than the one our scheme produced for the one-dimensional equations, and increasing the number of basis functions will take significantly more computer time and memory. Even for the reactionless heat equation it takes a few minutes to produce an 8 basis approximation for a solution at time 100. On the other hand, the system of ODEs (3.52) with initial conditions (3.54) could be solved analytically because (3.52) is an uncoupled system. Once we add our nonlinear reaction terms, it will become impossible to uncouple the ODEs produced by the Galerkin method, and we will normally be required to use numerical methods to solve the system.

Example 5: The Gray-Scott Model

This system mentioned in (2.8) will be the focus of our search for patterns. It is a variant of the autocatalytic Selkov model of glycolysis, due to Gray and Scott([5],[21]).

$$\begin{aligned} u_t &= d_u \Delta u - uv^2 + F(1-u) \\ v_t &= d_v \Delta v + uv^2 - (F+k)v \end{aligned} \quad (3.55)$$

In the model, u and v represent the concentrations of chemicals U and V, in standard chemistry notation u=[U] and v=[V]. The rate of change of the concentrations with respect to time is determined both by diffusion, modeled by the Laplacian of the concentrations with respect to space, and the chemical reaction rates which depend on the concentrations of the other chemicals at any point. The diffusion coefficients, given by d_u and d_v, are multiplied by the diffusion terms for each concentration. The reactions involved in the Gray-Scott model can be simplified to the following reactions:

$$U + 2V \rightarrow 3V \qquad (3.56)$$

$$V \rightarrow P. \qquad (3.57)$$

Both reactions proceed in only one direction. Reaction (3.56) proceeds at a rate proportional to $[U][V]^2=uv^2$ and acts to decrease the concentration of chemical U and increase the concentration of chemical V. Reaction (3.57) converts V to the inert product P at a rate of $k[V]$. F is a non-dimensionalized feed rate. We have four parameters for this system, F and k, the reaction parameters, and d_u and d_v, the diffusion coefficients.

In 1993, John Pearson from Los Alamos National Laboratories published results from finite difference simulations of the Gray-Scott model [18]. He used a mesh of 256 by 256 grid points and simulated solutions for 200,000 time steps. With the powerful supercomputers at Los Alamos National Laboratories, he was able to produce some very interesting patterns, from oscillating labyrinthine stripes, to stable hexagonal patterns of points, to dividing chemical rings resembling dividing cells. We have found some evidence of these remarkable patterns using the Galerkin method on much smaller computers for much smaller amounts of time.

As in Pearson's simulations we start with diffusion coefficients of d_u=2 x 10^{-5} and d_v=10^{-5}, but unlike his 1993 paper, we show patterns that occur with different diffusion coefficients as well. Our simulations take place on the region Ω=[-1,1]x[-1,1] with periodic boundary conditions. We choose the initial condition as a small perturbation from the homogeneous equilibrium point (u_0,v_0)=(1,0). Unlike Pearson's step function perturbations, our perturbations will be smooth. Explicitly, the perturbations will be Gaussian spots (a depression from 1 for u and a raised impression with a base at 0 for v). The following is an example of an initial condition with a maximum perturbation magnitude of 1/16 and centered at $(x,y)=(h,k)$ for both chemical u and chemical v:

$$u(x,y,0) = 1 - \frac{1}{16} e^{-20((x-h)^2+(y-k)^2)},$$
$$v(x,y,0) = 0 + \frac{1}{16} e^{-20((x+h)^2+(y-k)^2)}. \qquad (3.58)$$

A homogeneous equilibrium point is a point at which the concentrations are equal throughout the domain and are not changing with time. Later we explain how we know (1,0) is a homogeneous equilibrium point and we show that it is linearly stable for all reaction and diffusion parameter values. For now it suffices to know that at a stable equilibrium point the concentrations of all the chemicals in the system will remain constant over the entire domain, and will return to that constant state when slightly perturbed until some outside force acts to push the concentrations far enough away from their equilibrium values.

The purpose of using initial conditions that are small perturbations from homogeneous equilibrium points is to mimic the embryo's transition from a stable, patternless, homogenous

state to a state where stable, nonhomogeneous patterns exist. One of the great breakthroughs of Turing's work in morphogenesis was that he showed examples of homogeneous steady-state solutions that appear stable when only the chemical reactions are modeled, but become unstable when diffusion is considered. The system could then move to another steady-state solution, which may be nonhomogeneous [25]. Thus a chemical system initially in a state of homogeneous equilibrium could take on a variety of patterns, dependant on the reaction and diffusion parameters, as a small perturbation of the initial homogeneous state propagates throughout the domain. Murray explains this phenomenon of diffusion driven (or Turing) instability found in reaction-diffusion equations more thoroughly in chapter 14 of his text *Mathematical Biology* [15].

Our solution template is, as in the heat equation:

$$\tilde{u}(x,y,t) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y) \qquad (x,y)\in\Omega=[-1,1]\times[-1,1], t\in\mathbf{R}^{+}$$

$$\tilde{v}(x,y,t) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} b_{mn}(t)\phi_{mn}(x,y), \qquad (3.59)$$

where $\phi_{mn}(x,y)$ is given by (3.47). We will demonstrate the Galerkin procedure for the first equaition in (3.55). The procedure is identical for the second equation. Substituting (3.59) into the linear operator defined by the first equation in (3.55) we have

$$L(\tilde{u}) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a'_{mn}(t)\phi_{mn}(x,y) + d_u \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\pi^2(m^2+n^2)\phi_{mn}(x,y)$$
$$+ \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y)\left(\sum_{m1=-\infty}^{\infty}\sum_{n1=-\infty}^{\infty} b_{mn}(t)\phi_{mn}(x,y)\right)^2 - F\left(1-\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y)\right) = 0.$$

Using the weak characterization of 0 by setting the inner product with each basis function ϕ_{pq} equal to zero, we are left with equations of the form:

$$\begin{aligned}
&\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a'_{mn}(t)(\phi_{mn}(x,y),\phi_{pq}(x,y)) = \\
&-d_u \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\pi^2(m^2+n^2)(\phi_{mn}(x,y),\phi_{pq}(x,y)) - \\
&\left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y)\left(\sum_{m1=-\infty}^{\infty}\sum_{n1=-\infty}^{\infty} b_{m1n1}(t)\phi_{m1n1}(x,y)\right)^2,\phi_{pq}(x,y)\right) + \\
&(F,\phi_{pq}(x,y)) - F\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)(\phi_{mn}(x,y),\phi_{pq}(x,y))
\end{aligned} \tag{3.60}$$

The most complicated part of this expression contains the nonlinear terms that come from substituting the template solution into uv^2. Simplifying this expression, as we did with the heat equation, we have:

$$(uv^2,\phi_{pq}(x,y))) = \left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)\phi_{mn}(x,y)\left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} b_{mn}(t)\phi_{mn}(x,y)\right)^2,\phi_{pq}(x,y)\right) =$$

$$\sum_{m1=-\infty}^{\infty}\sum_{n1=-\infty}^{\infty}\sum_{m2=-\infty}^{\infty}\sum_{n2=-\infty}^{\infty}\sum_{m3=-\infty}^{\infty}\sum_{n3=-\infty}^{\infty} a_{m1n1}b_{m2n2}b_{m3n3}(\phi_{m1n1}(x,y)\phi_{m2n2}(x,y)\phi_{m3n3}(x,y),\phi_{pq}(x,y)).$$

Note the inner product inside the nonlinear term simplifies to:

$$(\phi_{m1n1}(x,y)\phi_{m2n2}(x,y)\phi_{m3n3}(x,y),\phi_{pq}(x,y)) = \begin{cases} 4, & (m1+m2+m3+p=0)\,and\,(n1+n2+n3+q=0), \\ 0, & (m1+m2+m3+p\neq 0)\,or\,(n1+n2+n3+q\neq 0). \end{cases}$$

(3.61)

If we truncate the infinite sums keeping only N terms, we will be left with $(2N+1)^2$ equations in $(2N+1)^2$ unknowns. Denoting the sum $\sum_{m1=-N}^{N}\sum_{n1=-N}^{N}\sum_{m2=-N}^{N}\sum_{n2=-N}^{N}\sum_{m3=-N}^{N}\sum_{n3=-N}^{N}$ by $\sum_{M(N)}^{N}$, we can write (3.60) in terms of ODEs of the form:

$$\begin{aligned}
&\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a'_{mn}(t)(\phi_{mn}(x,y),\phi_{pq}(x,y)) = \\
&\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(t)[d_u\pi^2(-m^2-n^2)-F](\phi_{mn}(x,y),\phi_{pq}(x,y)) - \\
&\sum_{M(N)} a_{m1n1}b_{m2n2}b_{m3n3}(\phi_{m1n1}(x,y)\phi_{m2n2}(x,y)\phi_{m3n3}(x,y),\phi_{pq}(x,y)) + \\
&(F,\phi_{pq}(x,y))
\end{aligned} \tag{3.62}$$

Using similar reasoning, we can write the second equation in (3.55) as:

$$\begin{aligned}
&\sum_{m=-N}^{N}\sum_{n=-N}^{N} b'_{mn}(t)(\phi_{mn}(x,y),\phi_{pq}(x,y)) = \\
&\sum_{m=-N}^{N}\sum_{n=-N}^{N} a_{mn}(t)[d_u\pi^2(-m^2-n^2)-F-k](\phi_{mn}(x,y),\phi_{pq}(x,y)) + \\
&\sum_{M(N)} a_{m1n1}b_{m2n2}b_{m3n3}(\phi_{m1n1}(x,y)\phi_{m2n2}(x,y)\phi_{m3n3}(x,y),\phi_{pq}(x,y)).
\end{aligned} \tag{3.63}$$

In this form (3.62) and (3.63), given appropriate initial data, can be solved numerically on a computer package such as MATLAB. This is made easier by using (3.61) which relies on the orthogonality of the basis functions to calculate the large inner product. We can also calculate the inner products ahead of time and store them to be referenced for particular index values. Since the time derivative is alone on the left side, we can use MATLAB's built in ODE solvers to calculate the values at a given time. See the appendix for the programs used in calculating the results.

4. The Gray-Scott Model

The Galerkin Spectral method and our MATLAB programs provide us with a way to produce numerical solutions of the Gray-Scott model for given parameter values. We will now analyze the model to determine where in parameter space we might begin looking for patterns. The first step is to examine its homogeneous equilibria. Recall that the Gray-Scott model is given by:

$$\begin{aligned} u_t &= d_u \Delta u - uv^2 + F(1-u) \\ v_t &= d_v \Delta v + uv^2 - (F+k)v \end{aligned} \qquad (4.1)$$

In the homogeneous state the concentrations do not change with respect to space and there will be no diffusion. In other words, the Laplacian terms that represent diffusion go to 0. The resulting equations are:

$$\begin{aligned} u' &= f(u,v) = -uv^2 + F(1-u) \\ v' &= g(u,v) = uv^2 - (F+k)v \end{aligned} \qquad (4.2)$$

It is interesting to note that for periodic or Neumann boundary conditions, (4.2) also results when considering only the 0^{th} order Galerkin approximation of (4.1).

Theorem 4.1: *When using periodic or Nuemann boundary conditions, the diffusionless equations describing the reaction kinetics in (4.2) are equivalent to the ODE's that result from the Galerkin method when choosing N=0.*

Proof: When $\phi_{m,n}(x,y) = e^{i\pi(mx+ny)}$ as for periodic boundary conditions, or else when $\phi_{m,n}(x,y) = \cos(\pi mx)\cos(\pi ny)$ as for Neumann boundary conditions, $\phi_{0,0}(x,y) = 1$. The solution template (3.46) with N=0 now takes the form:

$$\tilde{\mathbf{u}}(x,y,t) = \sum_{m=1}^{0}\sum_{n=1}^{0} \mathbf{a}_{m,n}(t)\phi_{m,n}(x,y) = \mathbf{a}_{0,0}(t)\phi_{0,0}(x,y) = \begin{cases} a_{0,0}(t) \\ b_{0,0}(t). \end{cases} \qquad (4.3)$$

Now when we substitute the right side of (4.3) into (4.1), we have

$$\begin{aligned} a'_{0,0} &= d_u \Delta a_{0,0} - a_{0,0}{b_{0,0}}^2 + F(1-a_{0,0}) = -a_{0,0}{b_{0,0}}^2 + F(1-a_{0,0}) \\ b'_{0,0} &= d_v \Delta b_{0,0} + a_{0,0}{b_{0,0}}^2 - (F+k)b_{0,0} = a_{0,0}{b_{0,0}}^2 - (F+k)b_{0,0}. \end{aligned} \qquad (4.4)$$

Since the coordinate functions $a_{0,0}$ and $b_{0,0}$ depend only on time, the diffusion terms in (4.3) that contain the Laplacian, a spatial differential operator, go to 0. Removing the subscripts we see that (4.4) is equivalent to (4.2):

$$a' = -ab^2 + F(1-a)$$
$$b' = ab^2 - (F+k)b. \qquad (4.5)$$

Thus (4.5), the system of equations that arises for the N=0 case, is equivalent to (4.2). Q.E.D.

The fact that the equilibrium points we are looking for are to be homogeneous allowed us to remove the diffusion term. The fact that they are equilibrium points means that they will not change over time, and thus the left side of (4.5) will be 0. In order to find these homogeneous equilibrium points we must now solve the nonlinear system:

$$-ab^2 + F(1-a) = 0$$
$$ab^2 - (F+k)b = 0. \qquad (4.6)$$

Solving for a in the first equation and substituting into the second equation in (4.6), we see that

$$b(b^2(F+k) - bF + F^2 + Fk) = 0, \qquad (4.7)$$

thus either b= 0, in which case a=1, or else b will be the root of the polynomial $b^2(F+k) - bF + F^2 + Fk$. We will show later that the equilibrium point, (a_0,b_0)=(1,0) will always be stable, even when diffusion is added. The b coordinates of the other equilibrium points are, by the quadratic formula,

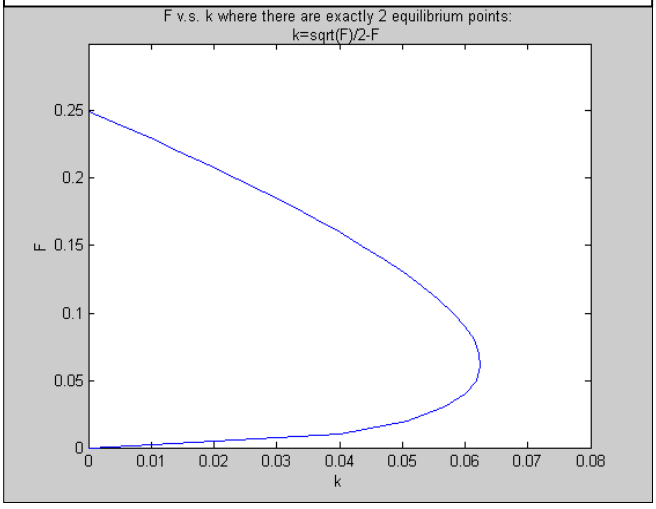
$$b = \frac{F \pm \sqrt{F^2 - 4(F+k)(F^2+Fk)}}{2(F+k)} \qquad (4.8)$$

and using the second equation in (4.6)

$$a = (F+k)/b. \qquad (4.9)$$

We will name the equilibrium point $(1,0)=(a_0,b_0)$. The equilibrium point with $b = \frac{F - \sqrt{F^2 - 4(F+k)(F^2+Fk)}}{2(F+k)}$, we will call (a_1,b_1), and the point with $b = \frac{F + \sqrt{F^2 - 4(F+k)(F^2+Fk)}}{2(F+k)}$ will be (a_2,b_2). The coordinate b_1 will always be

Figure 4.1: This curve gives the values of *F* and *k* where exactly 2 equilibria exist. Inside the region bounded by the curve and the y axis, there are 3 equilibria, and outside the only equilibrium point is



less than the coordinate b_2 and a_1 will always be greater than a_2. We will call (a_0,b_0)=(1,0) the independent equilibrium point and the other two dependent equilibrium points since they depend on F and k. The dependent equilibrium points will only exist when $k \le \sqrt{F}/2 - F$. In that region of F-k space, (4.2) has three equilibrium points (See fig 4.1). Elsewhere, (a_0,b_0) is the only equilibrium point. This is illustrated in Figure 4.1, which is also reproduced in [18]. Using calculus we can find the maximum k value where there will be exactly 2 equilibrium points:

$$k = \frac{\sqrt{F}}{2} - F$$

$$\frac{dk}{dF} = \frac{1}{4\sqrt{F}} - 1.$$

The critical point is at F=k=1/16=0.0625 as it appears to be in Figure 4.1. The maximum value of F for which the dependant equilibrium points can exist (see Figure 4.1) occurs at the larger value for which exactly two equilibrium points exist when k is equal to 0. The values of F for which there are exactly two equilibrium points with k=0 are given by $F = \frac{\sqrt{F}}{2}$. Solving for F we have F=0 and F=1/4 as indicated in Figure 4.1. Thus, region of F-k space on which the dependent homogeneous equilibrium points exist is bounded by

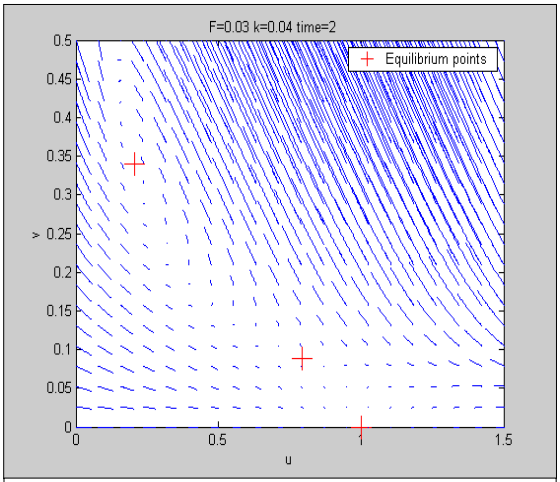


Figure 4.2 Field plot and equilibrium points of the ODE system (4.2) for F=0.03, k=0.04.

$$0 \le F \le 1/4$$
$$0 \le k \le 1/16. \qquad (4.10)$$

Figure 4.2 shows the equilibrium points and the vector field plot of the differential equation in the ab plane for F=0.03 and k=0.04. At these particular parameter values, there are three equilibrium points. Figures 4.3 and 4.4 show the dependence of the equilibrium points given by (4.8) and (4.9) on F and k as these parameters are respectively held constant. In figure 4.3, the equilibrium points approach each other from either side as k is increased. In Figure 4.4, the equilibrium points appear, traverse around an enclosed path, and vanish as F increases from $(1/4 - \sqrt{1/16 - k})^2$ to $(1/4 + \sqrt{1/16 - k})^2$.

The surface and the contour plot in Figure 4.5 show how the equilibrium points change with both F and k. Analyzing Figure 4.5 we can visualize several properties of the system:

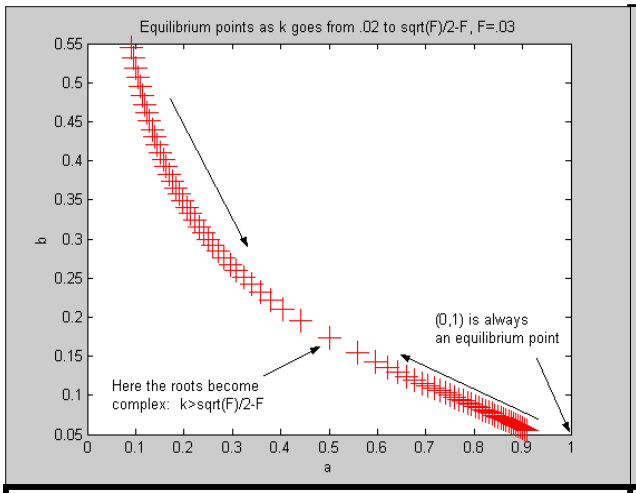


Figure 4.3: Equilibrium points as *k* varies and F is held constant.

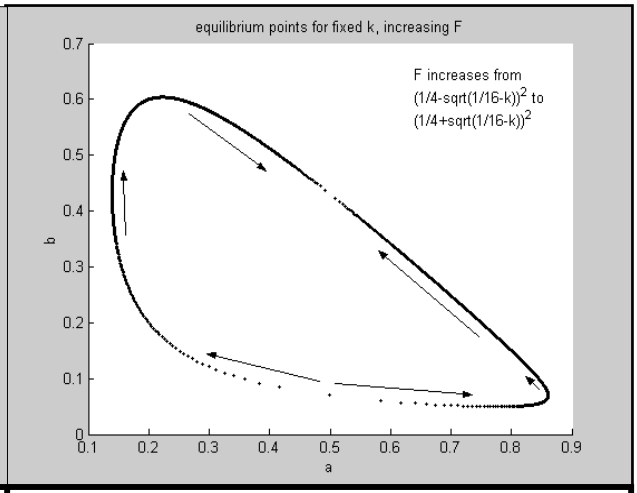


Figure 4.4: Equilibrium points as *F* is varied and *k* is held constant.

Theorem 4.2: *When there are exactly two equilibrium points, a=1/2.*

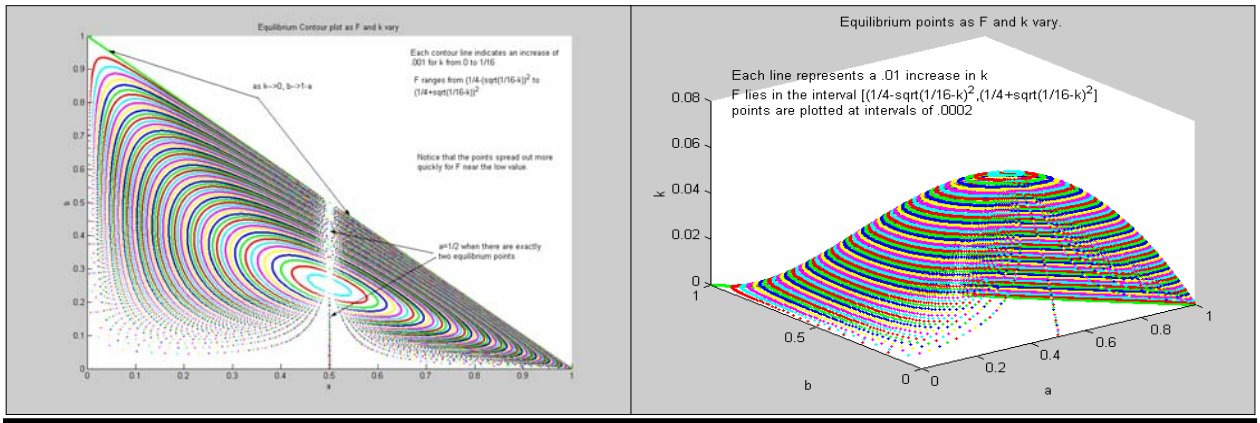


Figure 4.5: Visualizing the dependence of the equilibrium points on *F* and *k*.

Proof: When there are exactly two equilibrium points, the term under the radical in (4.8) is zero. This is true when $k = \frac{\sqrt{F}}{2} - F$. Thus we see that the equilibrium point's a coordinate is at:

$$a = \frac{F+k}{b} = \frac{(F+k)}{\left(\frac{F}{2(F+k)}\right)} = \frac{2(F+k)^2}{F} = \frac{2(F+\sqrt{F}/2 - F)^2}{F} = \frac{2(F/4)}{F} = 1/2.$$

Q.E.D.

Theorem 4.3: $\lim_{k\to 0}(a+b) = 1$.

Proof: As $k \to 0$ we see from (4.8) that $b \to \frac{F \pm \sqrt{F^2 - 4F(F^2)}}{2F} = 1/2 \pm \sqrt{1/4 - F}$ and from (4.9) that $a \to F/b$. Thus

$$\lim_{k\to 0} a = \frac{F}{1/2 \pm \sqrt{1/4 - F}} = \frac{F}{1/2 \pm \sqrt{1/4 - F}} \frac{1/2 \mp \sqrt{1/4 - F}}{1/2 \mp \sqrt{1/4 - F}} = \frac{F(1/2 \mp \sqrt{1/4 - F})}{1/4 - (1/4 - F)} =$$
$$1 - (1/2 \pm \sqrt{1/4 - F}) = 1 - \lim_{k\to 0} b,$$

or in other words $\lim_{k\to 0}(a+b) = 1$.

Q.E.D.

Now that we have a sense of the dependence of the equilibria on the parameters, we can ask whether or not they are stable. An equilibrium point of (4.2) is defined as stable if small perturbations from the equilibrium will eventually settle back to the equilibrium point. As is explained in depth in [10], p. 178-179, [22], p150, and in [15], p. 697-701, the stability of a nonlinear system can be determined from the eigenvalues of the matrix obtained by linearizing the system about the equilibrim point. Let $u=p$ and $v=q$ be an equilibrium point of the system, so that:

$$f(p,q) = g(p,q) = 0. \qquad (4.11)$$

We linearize the system about (p,q). That is, we take a Taylor series expansion of f and g for u and v very close to (p,q) and keep only the linear terms. We set $a=(p+\varepsilon)$ and $b=(q+\delta)$ where ε and δ are small, and find the Taylor series expansion about (p,q):

$$a_t = f(a,b) = f(p+\varepsilon, q+\delta) = f(p,q) + \varepsilon \tfrac{\partial f}{\partial a}(p,q) + \delta \tfrac{\partial f}{\partial b}(p,q) + \textit{higher order terms}$$
$$b_t = g(a,b) = g(p+\varepsilon, q+\delta) = g(p,q) + \varepsilon \tfrac{\partial g}{\partial a}(p,q) + \delta \tfrac{\partial g}{\partial b}(p,q) + \textit{higher order terms}.$$

(4.12)

Noticing that $a_t = \varepsilon_t$ and $b_t = \delta_t$, using (4.11), and ignoring the higher order terms of (4.12) because they are proportional to the small numbers ε and δ raised to the second degree and higher, we rewrite (4.12) as:

$$\begin{aligned} \varepsilon_t &= \varepsilon \tfrac{\partial f}{\partial a}(p,q) + \delta \tfrac{\partial f}{\partial b}(p,q) \\ \delta_t &= \varepsilon \tfrac{\partial g}{\partial a}(p,q) + \delta \tfrac{\partial g}{\partial b}(p,q). \end{aligned} \qquad (4.13)$$

The solutions to (4.13) depend on the eigenvalues of the matrix

$$J_{p,q} = \left(\begin{bmatrix} \dfrac{\partial f}{\partial a} & \dfrac{\partial f}{\partial b} \\ \dfrac{\partial g}{\partial a} & \dfrac{\partial g}{\partial b} \end{bmatrix} \right)_{p,q} \qquad (4.14)$$

which we will call the stability matrix. The notation indicates that it is evaluated at the equilibrium point, (p,q), where we would like to test stability.

The solutions to (4.13) will contain terms with the eigenvalues of J in the exponent. Thus if any of the eigenvalues of J has a positive real part, the perturbations δ and ε will increase without bound and the system is unstable. If J's eigenvalues have negative real parts, then the perturbations will decrease exponentially with time and the system is stable.

For the Gray-Scott reaction system (4.2), the stability matrix is given by:

$$J_{a_0,b_0} = \begin{bmatrix} -b^2 - F & -2ab \\ b^2 & ab - (F+k) \end{bmatrix}. \qquad (4.15)$$

From (4.15) we can see that the eigenvalues of $J_{p,q}$, and thus the linear stability of (4.2) at (p,q), depends on the equilibrium points and the parameters F and k. From (4.6), (4.8) and (4.9), we know that the equilibrium points themselves depend on the parameter values F and k, so the stability of the eigenvalues depends ultimately on F and k in our model.

Theorem 4.4: *The independent equilibrium point, $(a_0,b_0)=(1,0)$ is always stable for (4.2).*

Proof: At (a_0,b_0),

$$J_{a_0,b_0} = \begin{bmatrix} -F & 0 \\ 0 & -(F+k) \end{bmatrix}. \qquad (4.16)$$

For diagonal matrices, the eigenvalues are the diagonal entries, so (4.16) has eigenvalues of $-F$ and $-(F+k)$. Since the feed rate F and the rate constant k will always be positive number, the eigenvalues of (4.16) will always be negative for any valid choice of F and k.

Q.E.D.

Theorem 4.5: *The dependent equilibrium point,* (a_1,b_1) *is always unstable for all plausible values of F and k.*

Proof: At (a_1,b_1),

$$J_{a_1,b_1} = \begin{bmatrix} -b^2 - F & -2\left(\frac{F+k}{b}\right)b \\ b^2 & 2\left(\frac{F+k}{b}\right)b - (F+k) \end{bmatrix} = \begin{bmatrix} -b^2 - F & -2(F+k) \\ b^2 & F+k \end{bmatrix} \qquad (4.17)$$

The determinant of this matrix is

$$Det(J_{a_1b_1}) = (-b^2 - F)(F+k) + 2(F+k)b^2 = (F+k)(b^2 - F)$$

Since the entries of the matrix $J_{a_1b_1}$ are all real, the eigenvalues of the matrix will either be real or, if they are complex, complex conjugates of one another. If we can show that the determinant of $J_{a_1b_1}$ is negative, we will know that its eigenvalues are real and that one eigenvalue is positive, while the other eigenvalue is negative. That is because the determinant of a matrix is the product of its eigenvalues. The factor $(F+k)$ is always positive because F and k are both positive values. Our goal, therefore, is to show that the term $(b^2 - F)$ is always negative for all plausible values of F and k. Note here that from (4.10) that $0 \le F \le 1/4$ and $0 \le k \le 1/16$.

The equilibrium point's b value, b_1, is given by:

$$b_1 = \frac{F - \sqrt{F^2 - 4F(F+k)^2}}{2(F+k)}.$$

The term inside the radical must be between 0 and 1;

$$0 \le F^2 - 4F(F+k)^2 \le 1$$

since it must be positive for b_1 to be real and since $F^2 \le F \le 1/4 < 1$. It then follows that:

$$F^2 - 4(F+k)^2 \le F^2 - 4F(F+k)^2 \le \sqrt{F^2 - 4F(F+k)^2}\ .$$

The left inequality is true because $F \le 1/4 < 1$ and the right inequality is true because the term inside the radical is less than one. We rearrange the resulting inequality to get *b* on the left side:

$$F^2 - \sqrt{F^2 - 4F(F+k)^2} \le 4(F+k)^2$$

$$b = \frac{F^2 - \sqrt{F^2 - 4F(F+k)^2}}{2(F+k)} \le 2(F+k)$$

Now we manipulate this expression to get it into a more convenient form:

$$\frac{bF}{(F+k)} \le 2F$$

$$\frac{bF - F(F+k)}{(F+k)} - F \le 0. \qquad (4.18)$$

From the original polynomial (4.7), we can get an expression for b^2 in terms of *b*:

$$b^2 = \frac{bF - F(F+k)}{(F+k)}. \qquad (4.19)$$

Now using (4.18) and (4.19) we can show:

$$b^2 - F = \frac{bF - F(F+k)}{(F+k)} - F \le 0\,,$$

which means that the determinant of the matrix $J_{a_1 b_1}$ is negative. Therefore both its eigenvalues are real and one eigenvalue is positive, while the other is negative. The equilibrium point (a_1,b_1) will always be unstable for valid values of *F* and *k*.

Q.E.D.

Determining stability at the dependent equilibrium points (a_2,b_2) is slightly more complicated than for (a_1,b_1) and (a_0,b_0) since it has complex eigenvalues. Figure 4.6 is a plot of the

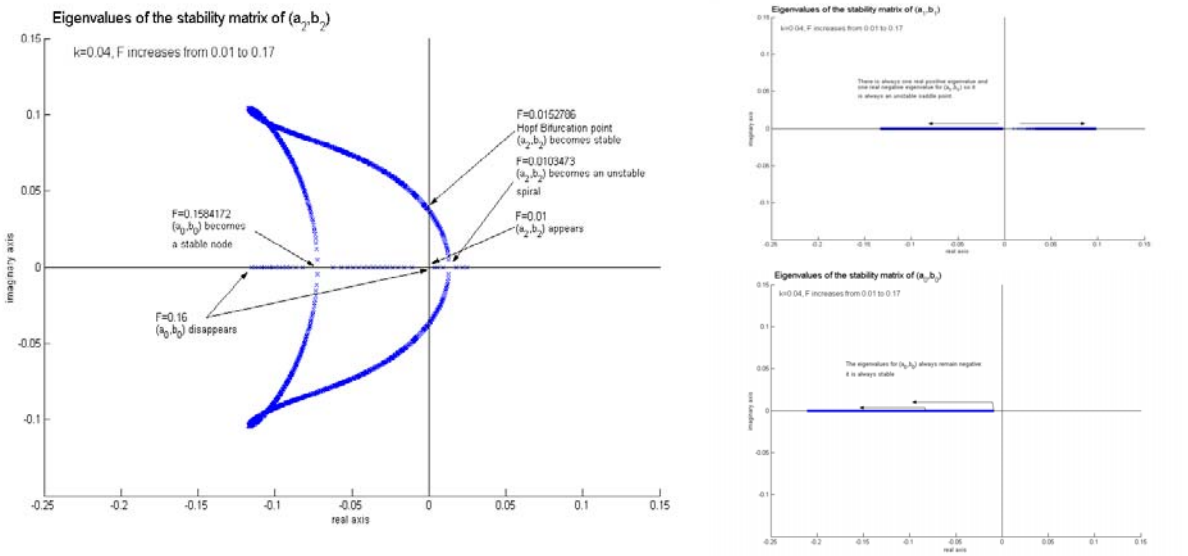
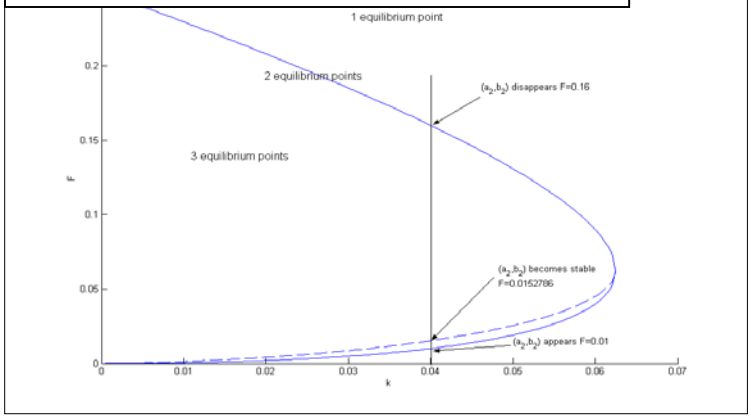


Fig. 4.6 Eigenvalues of the stability matrix for fixed *k* as *F* varies

eigenvalues of (4.15) in the complex plane evaluated at the equilibrium points as F varies between 0.01 and 0.17 and k is held at 0.4. The eigenvalues of the dependent equilibrium point (a_2,b_2) are the complex conjugates of one another, and they cross into the left half plane with nonzero imaginary components. When this occurs, a family of periodic solutions appear surrounding the equilibrium point. This phenomenon is known as Hopf bifurcation and is discussed in detail in [15], p 706-719. The dotted line in Figure 4.7 indicates where in F-k space these Hopf bifurcation points will occur. Notice the behavior of the eigenvalues in Figure 4.6 in the different portions of the vertical line at k=0.04. When we add diffusion we will search for patterns around this Hopf bifurcation line in parameter space reasoning that these periodic solutions in the N=0 case

Presence of equilibria in the diffusionless Gray-Scott model. **Fig 4.7**



might give rise to nonhomogeneous steady-state solutions and periodic orbits when diffusion is added. Figure 4.7 is a reproduction of a similar diagram found in [18].

From Figure 4.6 we can also see that, as we established earlier, (a_0,b_0) is always stable, and (a_1,b_1) is always unstable. The fact that (a_1,b_1) has one negative and one positive eigenvalue also means that it is a saddle point. Figure 4.8 shows the phase plane of (4.2) for F=0.011, k=0.04. These parameter values lie between the dotted line and the lower solid line in Figure 4.7. At these values we will therefore have three equilibrium points, two of which will be unstable. The small cross in Figure 4.8 indicates the location of (a_1,b_1), and the circles represent the other equilibria. Since (a_1,b_1) is a saddle point, it must have an unstable and a stable manifold. That is, there must be a curve along which solutions are attracted and a curve along which solutions are repelled. These are shown in the figure, and they help us understand why solutions arising from initial values very close to the stable equilibrium point (a_0,b_0) sometimes follow roundabout trajectories before they finally settle into (a_0,b_0).

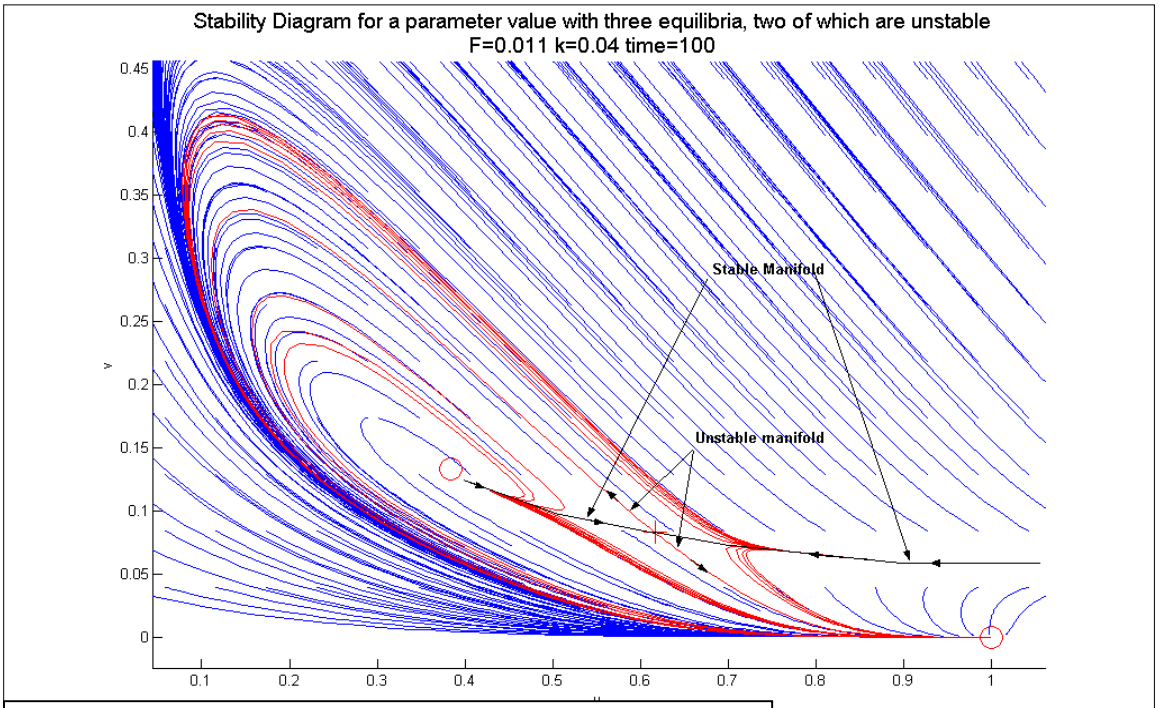


Fig 4.8 Stable and unstable manifolds of the saddle point (a_1,b_1)

Now that we understand the behavior of the system without diffusion, we will try to get a sense of what happens once diffusion is added. We will first derive the conditions for linear

stability in the system once the diffusion term is included in the analysis. Consider the general two species reaction-diffusion equation:

$$\begin{aligned} u_t &= d_u \Delta u + f(u,v), \\ v_t &= d_v \Delta v + g(u,v). \end{aligned} \tag{4.20}$$

We apply the standard stability analysis of partial differential equations by analyzing the behavior of solutions of (4.20) in a small neighborhood of (u_0,v_0). This procedure is similar to the stability analysis outlined above. To that end, we make the substitutions

$$\begin{aligned} U &= u_0 + \varepsilon u, \\ V &= v_0 + \varepsilon v. \end{aligned}$$

Linearizing f and g near the equilibrium point (u_0,v_0) by using a Taylor series expansion about (u_0,v_0), we find that for small ε:

$$f(U,V) = f(u_0+\varepsilon, v_0+\varepsilon) = f(u_0,v_0) + \varepsilon\left(\frac{\partial f}{\partial u}|_{(u_0,v_0)}\ u + \frac{\partial f}{\partial v}|_{(u_0,v_0)}\ v\right) + O(\varepsilon^2) + \ldots,$$

$$g(U,V) = g(u_0+\varepsilon, v_0+\varepsilon) = g(u_0,v_0) + \varepsilon\left(\frac{\partial g}{\partial u}|_{(u_0,v_0)}\ u + \frac{\partial g}{\partial v}|_{(u_0,v_0)}\ v\right) + O(\varepsilon^2) + \ldots .$$

But since we are concerned only with very small ε the $O(\varepsilon^2)$ terms can be ignored. Also since (u_0,v_0) is an equilibrium point, the constant terms, $f(u_0,v_0) = g(u_0,v_0)$ are zero. We are thus left with:

$$\begin{aligned} f(U,V) &\approx \varepsilon\left(\frac{\partial f}{\partial u}|_{(u_0,v_0)}\ u + \frac{\partial f}{\partial v}|_{(u_0,v_0)}\ v\right) \\ g(U,V) &\approx \varepsilon\left(\frac{\partial g}{\partial u}|_{(u_0,v_0)}\ u + \frac{\partial g}{\partial v}|_{(u_0,v_0)}\ v\right) \end{aligned} = \varepsilon J_{u_0,v_0}\begin{pmatrix} u \\ v \end{pmatrix}$$

where J_{u_0,v_0} is the stability matrix

$$J = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \equiv \begin{bmatrix} \dfrac{df}{du}|_{(u_0,v_0)} & \dfrac{df}{dv}|_{(u_0,v_0)} \\ \dfrac{dg}{du}|_{(u_0,v_0)} & \dfrac{dg}{dv}|_{(u_0,v_0)} \end{bmatrix} \tag{4.21}$$

of the functions f and g evaluated at (u_0,v_0). We also note that

$$\begin{aligned} U_t &= \varepsilon u_t \\ V_t &= \varepsilon v_t \end{aligned} \quad \text{and} \quad \begin{aligned} \Delta U &= \varepsilon \Delta u \\ \Delta V &= \varepsilon \Delta v \end{aligned}.$$

Substituting U and V into (4.20), dividing out by ε, and ignoring the remaining terms that depend on ε, we see that near equilibrium u and v satisfy the following linear partial differential equation:

$$\begin{aligned} u_t &= d_u \Delta u + au + bv, \\ v_t &= d_v \Delta v + cu + dv. \end{aligned} \tag{4.22}$$

Now let (u_0,v_0) be a linearly stable equilibrium point for the diffusionless system

$$\begin{aligned} u_t &= f(u,v), \\ v_t &= g(u,v). \end{aligned} \tag{4.23}$$

Note that the linearization of (4.23) about (u_0,v_0) is

$$\begin{pmatrix} u_t \\ v_t \end{pmatrix} = J_{u_0,v_0} \begin{pmatrix} u \\ v \end{pmatrix}.$$

The assumption that (u_0,v_0) is linearly stable is equivalent to assuming that the real parts of the eigenvalues of J_{u_0,v_0} are always negative. Let λ_1 and λ_2 be the eigenvalues of J_{u_0,v_0}. The fact that they are always negative means that:

$$a + d = Tr(J) = \lambda_1 + \lambda_2 < 0, \tag{4.24}$$

and

$$ad - bc = Det(J) = \lambda_1 \lambda_2 > 0. \tag{4.25}$$

We seek solutions to (4.14) of the form

$$\begin{aligned} u(x,y,t) &= Ae^{\lambda t}\varphi_{ij}, \\ v(x,y,t) &= Be^{\lambda t}\varphi_{ij}, \end{aligned} \tag{4.26}$$

where $\varphi_{ij}(x,y)$ is an eigenfunction of the laplacian operator Δ, i.e., $\Delta\, \varphi_{ij}(x,y) = \delta_{ij}\, \varphi_{ij}(x,y)$ where δ_{ij} is a negative scalar depending on the particular basis function used. Substituting (4.26) into (4.20) yields

$$\begin{aligned} \lambda A e^{\lambda t}\varphi_{ij} &= \delta_{ij} A d_u e^{\lambda t}\varphi_{ij} + aAe^{\lambda t}\varphi_{ij} + bBe^{\lambda t}\varphi_{ij} \\ \lambda B e^{\lambda t}\varphi_{ij} &= \delta_{ij} B d_v e^{\lambda t}\varphi_{ij} + cAe^{\lambda t}\varphi_{ij} + dBe^{\lambda t}\varphi_{ij}. \end{aligned}$$

After dividing by $e^{\lambda t}\varphi_{ij}$ we have:

$$\lambda\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} a+\delta_{ij}d_u & b \\ c & d+\delta_{ij}d_v \end{bmatrix}\begin{bmatrix} A \\ B \end{bmatrix}.$$

So for solutions described by (4.26) to exist, λ must be an eigenvalue of the matrix

$$D_{ij} = \begin{bmatrix} a+\delta_{ij}d_u & b \\ c & d+\delta_{ij}d_v \end{bmatrix}. \qquad (4.27)$$

In order for (4.26) to be a linearly stable solution of (4.20), the eigenvalues λ_a and λ_b of (4.27) must all have negative real parts. Using the properties of determinants and trace, this means that

$$\delta_{ij}{}^2(d_u d_v) + \delta_{ij}(dd_u + ad_v) + (ad - bc) = Det(D_{ij}) > 0, \qquad (4.28)$$

and

$$a + d + \delta_{ij}(d_u + d_v) = Tr(D_{ij}) = \lambda_a + \lambda_b < 0. \qquad (4.29)$$

These are necessary and sufficient conditions for stability, whether or not the eigenvalues are complex.

One phenomenon that is of interest in pattern formation in reaction-diffusion systems is that of diffusion driven instability. A homogeneous equilibrium point that exhibits diffusion driven instability will be stable for the diffusionless system (4.23), but will become unstable once diffusion is accounted for. Diffusion is usually a process that contributes to stability in a system by decreasing concentrations where they are unusually high, and increasing concentrations where they are lower than the surrounding regions. In our reaction-diffusion system, we have found a region of parameter values for which diffusion breaks the linear stability of a system. We have found evidence of stable periodic and nonhomogeneous steady-state solutions arising in this parameter region as the systems proceed away from the unstable homogeneous equilibrium. Murray and others ([15],[17]) have shown in other reaction-diffusion systems that pattern

formation in the form of periodic solutions and nonhomogeneous steady-state solutions can occur for parameter values regions where diffusion driven instability has been shown to exist. Perhaps the instability brought on by the normally restorative process of diffusion combined with the fact that the system has several stable homogeneous solutions without diffusion is enough to make the system unstable enough to drive solutions away from the homogeneous state, while remaining stable enough to find the nonhomogeneous steady-state and periodic solutions in between homogeneous equilibria that we call patterns. The following result is a necessary condition for diffusion driven instability.

Theorem 4.6: *In order for diffusion driven instability to occur in a two species reaction-diffusion equation, the diffusion constants must be unequal. In other words, in order for a homogeneous equilibrium solution (u_1,v_1) to be linearly stable for the diffusionless system (4.23), and linearly unstable for the full system (4.24),* $d_u \neq d_v$.

Proof: In order for the homogeneous equilibrium solution (u_1,v_1) to be linearly stable for (4.23), conditions (4.24) and (4.25) must hold, i.e. $a+d<0$ and $ad-bc>0$. If we let the diffusion constants equal one another, $d_u=d_v=d_{uv}$, then the left side of condition (4.28) becomes

$$\delta_{ij}{}^2 d_{uv}{}^2 + \delta_{ij} d_{uv}(a+d) + (ad-bc).$$

The first term in this expression will always be positive. The second term is also positive since δ_{ij} is a negative number, the diffusion constant is a positive number, and ($a+d$) is negative. The last term is also positive, so (4.28) always holds.

With equal diffusion constants, the left side of (4.29) becomes:

$$a + d + 2\delta_{ij} d_{uv},$$

which by similar reasoning is always negative causing (4.29) to hold for all parameters. Thus (u_1,v_1) always satisfies sufficient conditions for stability in the full system (4.22) and will always be stable. There can be no diffusion driven instability.

Q.E.D.

We will now see how this condition can be applied to the Gray-Scott model to find regions of diffusion driven instability. The first thing we would like to show is that the homogeneous equilibrium point (a_0,b_0) remains linearly stable for any choice of *F, k* and diffusion constant d_u and d_v.

Theorem 4.7: *The homogeneous equilibrium solution to the Gray-Scott equations (4.1) at* $(u_0,v_0)= (1,0)$ *is linearly stable for all feasible values of F, k,* d_u *and* d_v.

Proof: First of all, we know that $(u_0,v_0)=(1,0)$ is a homogeneous solution for (4.1), because the point $(a_0,b_0)=(1,0)$ is an equilibrium point of the diffusionless system. In fact, Theorem 4.4 states that (a_0,b_0) is linearly stable for all *F* and *k*. For the Gray-Scott reaction-diffusion equation, the matrix *J* from (4.21) is

$$J = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} -v^2 - F & -2uv \\ v^2 & 2uv - (F+k) \end{bmatrix},$$

and the matrix D_{ij} is given by

$$D_{ij} = \begin{bmatrix} a + \delta_{ij} d_u & b \\ c & d + \delta_{ij} d_v \end{bmatrix} = \begin{bmatrix} -v^2 - F + \delta_{ij} d_u & -2uv \\ v^2 & 2uv - (F+k) + \delta_{ij} d_v \end{bmatrix}. \quad (4.30)$$

Substituting the values $(u_0,v_0)=(1,0)$ into (4.30) we have

$$D_{ij} = \begin{bmatrix} a + \delta_{ij} d_u & b \\ c & d + \delta_{ij} d_v \end{bmatrix} = \begin{bmatrix} -F + \delta_{ij} d_u & 0 \\ 0 & -(F+k) + \delta_{ij} d_v \end{bmatrix}. \quad (4.31)$$

The left side of condition (4.28) becomes:

$$\delta_{ij}{}^2 d_u d_v - \delta_{ij}(F d_v + (F+k) d_u) + F(F+k).$$

Since δ_{ij} is always negative and the reaction and diffusion parameters are always positive, this expression is always positive and (4.28) always holds.

For (a_0,b_0), the left side of condition (4.29) becomes:

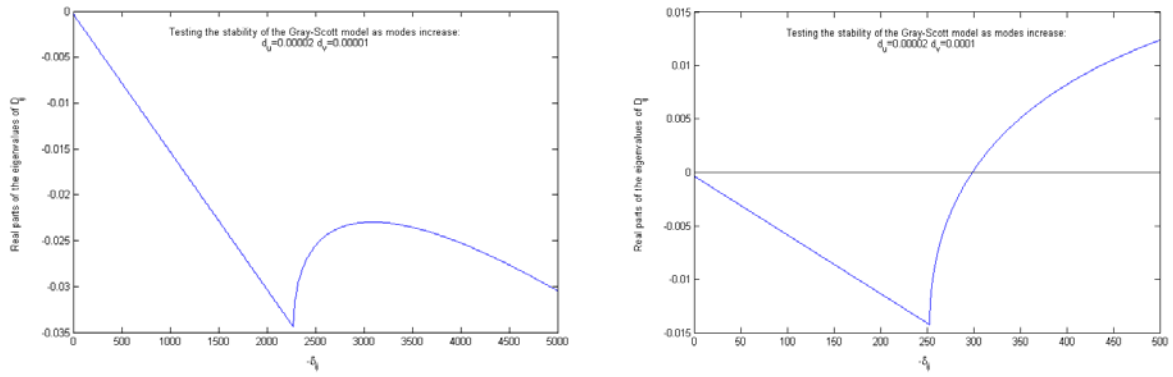
$$-(2F+k) + \delta_{ij}(d_u + d_v).$$

This expression is always negative, and thus condition (4.29) holds. Since both conditions hold, the equilibrium point (a_0,b_0) is stable for all valid reaction and diffusion parameter values.

Q.E.D.

For (u_2,v_2)=(a_2,b_2), the other equilibrium point that can be linearly stable for the diffusionless system, the conditions for stability in the full system, (4.28) and (4.29), may not always be satisfied. In order to examine the stability of the full system with diffusion, we will look at the real parts of the eigenvalues of the stability matrix. In the Hopf bifurcation region, when (a_2,b_2) first becomes stable, the eigenvalues will always be complex conjugates of one another. Thus the real part of one eigenvalue will be equal to the other. If this value is negative the system is stable, and if it is positive the system is unstable. Figure 4.9 is a plot of the real part of one eigenvalue of the matrix D_{ij} (4.30) for $F=0.0154$, $k=0.04$, $d_v=0.00001$, and d_u values of 0.0001 and 0.00002 for the homogeneous equilibrium point (a_2,b_2) versus δ_{ij}.

Figure 4.9 Real parts of the eigenvalues of matrix (4.30) as $-\delta_{ij}$ is increased. The plot on the left has d_u=0.00002 and the plot on the right has d_u=0.0001. All other parameters are equal at F=0.0154, k=0.04, and d_v=0.00001.



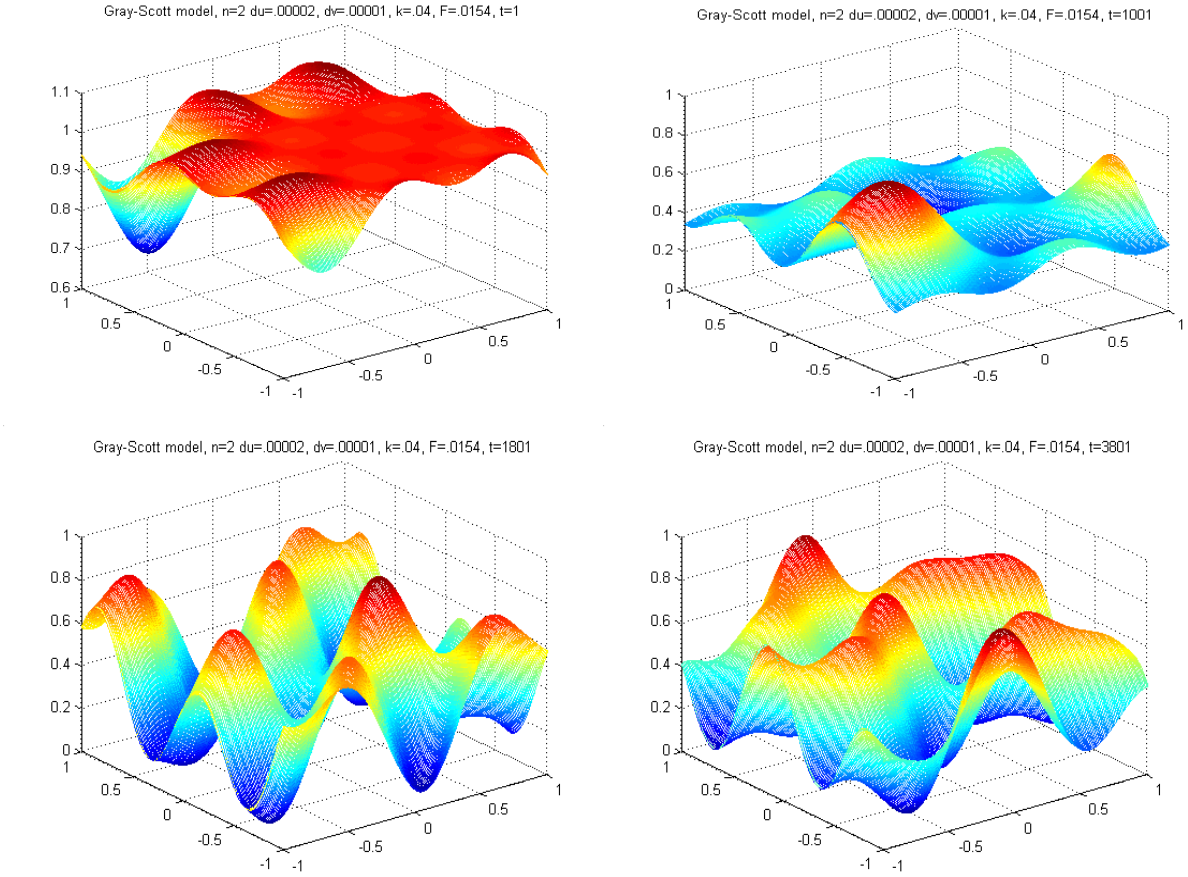
The plot on the left shows that the real part of the greatest eigenvalue of the stability matrix never becomes positive, and thus diffusion driven instability does not occur. On the right, we see that the real parts of the initially stable point start negative, but suddenly one real part breaks up and becomes positive. At this point the eigenvalues are no longer complex and the real part of the other eigenvalue, which remains negative, is not shown. The system becomes linearly unstable and diffusion driven instability exists. We will look at patterns for both of these diffusion parameters, and see what effect this type of instability seems to have on the system. This analysis seems to indicate at least initially that the magnitude of the difference between the diffusion parameters of the morphogens is a primary factor in diffusion driven instability.

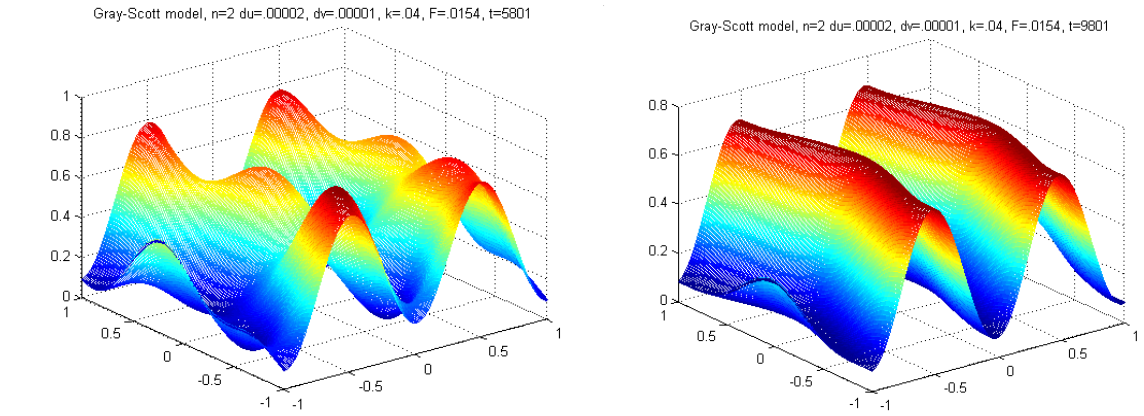
Murray and others have examined diffusion driven instability searching for Turing patterns. Turing's original idea was that patterns could be driven by the diffusion terms and not the reaction terms [15],[17],[25]. We have found some indications of pattern formation inside an envelope of parameter values where diffusion driven instability can exist for certain modes and diffusion coefficients.

Results

This section displays some of the patterns that we observed for various parameter values. The values of the reaction parameters that we used, F=0.0154 and k=0.04, fall above the Hopf bifurcation curve of Figure 4.7. With N=0, therefore $(a_2,b_2) \approx (0.2748, 0.2016)$ would be a stable, spiral equilibrium point. Figure 5.1 shows snapshots of our simulated solution of u at these parameter values with N=2. Recall that our diffusion constants are d_u=2 x 10^{-5} and d_v=10^{-5}. Our initial condition is a depression centered at (x,y)=(-0.65, 0.65). By the time t reaches 5801, we can see the form of a morphogen wave which becomes fully established by time 9801.

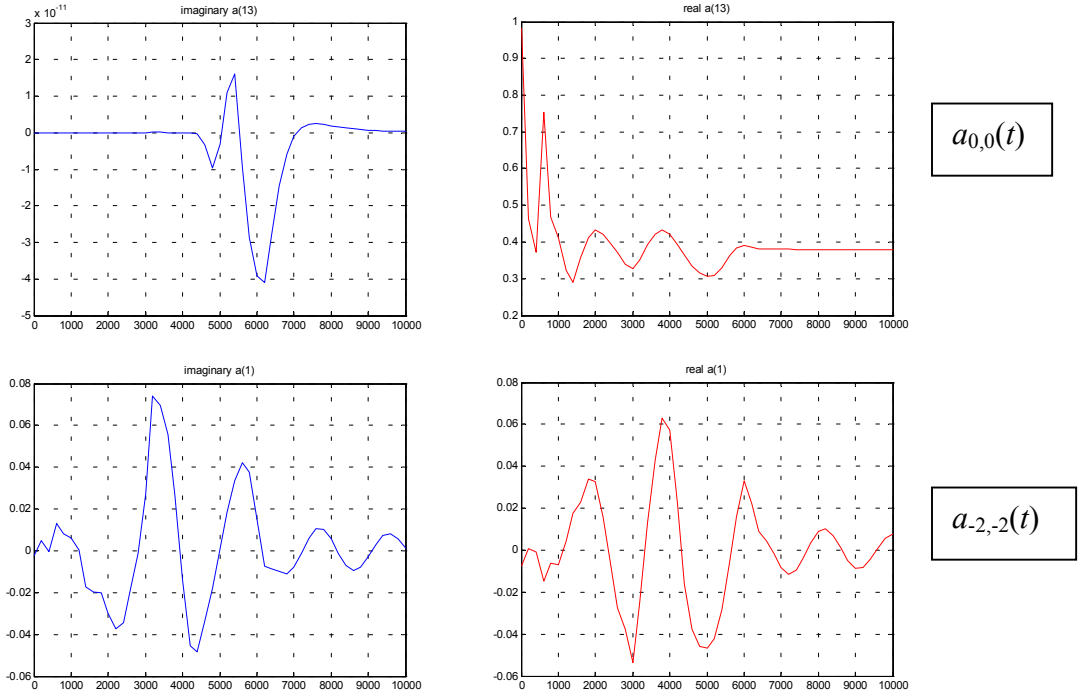
Figure 5.1 Simulated solution for *u* of the Gray-Scott model with *N*=2.

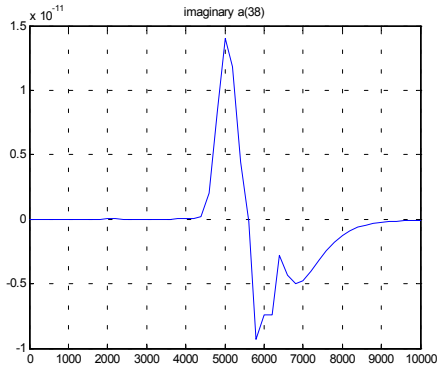


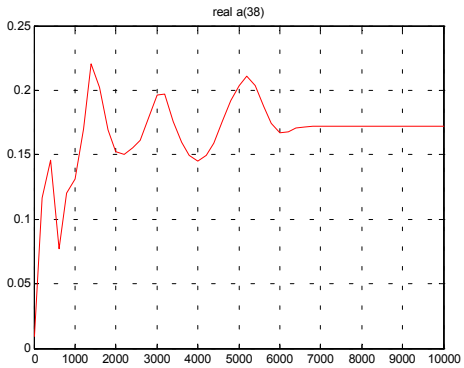


In Figure 5.2 we have plots of a few of the coordinate functions of time from the solution template (3.59), the $a_{i,j}(t)$ and $b_{i,j}(t)$ versus time for the solution from Figure 5.1. Notice that the imaginary parts for $a_{0,0}(t)$ and $b_{0,0}(t)$ are essentially 0. Near time 6000 we see that the coordinate functions for the 0,0 modes, which we will call the principal coordinate functions, seem to

Figure 5.2 - Several coordinate functions of time for F=0.0154, k=0.04, N=2. Real parts are on the right and imaginary parts are on the left. The captions on the individual pictures refer to the index of the a vector in the MATLAB program GAL_PEARSON2D (see appendix).







$b_{0,0}(t)$

achieve a steady state. The values they obtain are not far from the spiral homogeneous equilibrium of the N=0 problem $(a_2,b_2) \approx (0.2748, 0.2016)$. The coordinate function $a_{1,1}(t)$ begins to oscillate steadily around this same time. This time also corresponds to the time at which the wave begins to appear in Figure 5.1.

Figure 5.3 shows what happens when we increase N. We plot u for the same parameter values, except that N=4 for times up to 1000. This time we have placed the initial perturbation in the center, but this does not appreciably affect the results. The perturbation spreads to the boundaries and from the site of the perturbation a wave seems to be emanating by time 1000. Figure 5.4 shows the contour plots of u with N=4, this time with the initial perturbation off center again. As time increases past 1000, the circular waves emanating from the site of the initial perturbation seem to take on transverse wavelike motion, similar to those we saw in Figure 5.1 but not as clearly defined.

As N is increased, we are able to use more modes to approximate the solution and thus we are able to capture more detail and complexity. If waves form in the true solution for these parameter values, they may not look like those we obtained for N=2 and N=4. However, if the true solutions do indeed have waves, these low level approximations could prove useful in predicting the formation of waves for a particular parameter set. Comparing the solutions near

time 1000 for n=2 and n=4, there is little resemblance. We cannot be confident that our simulated solution is close to the true solution until we can increase the number of modes without any significant change in the simulated solution. So far N=4 is the highest number of modes we have been able to use with MATLAB. If we were to use Neumann boundary conditions and a smaller region, we might be able to try more modes.

Figure 5.3 Surface Plots of N=4 with F=0.0154, k=0.04, for times up to 1000.

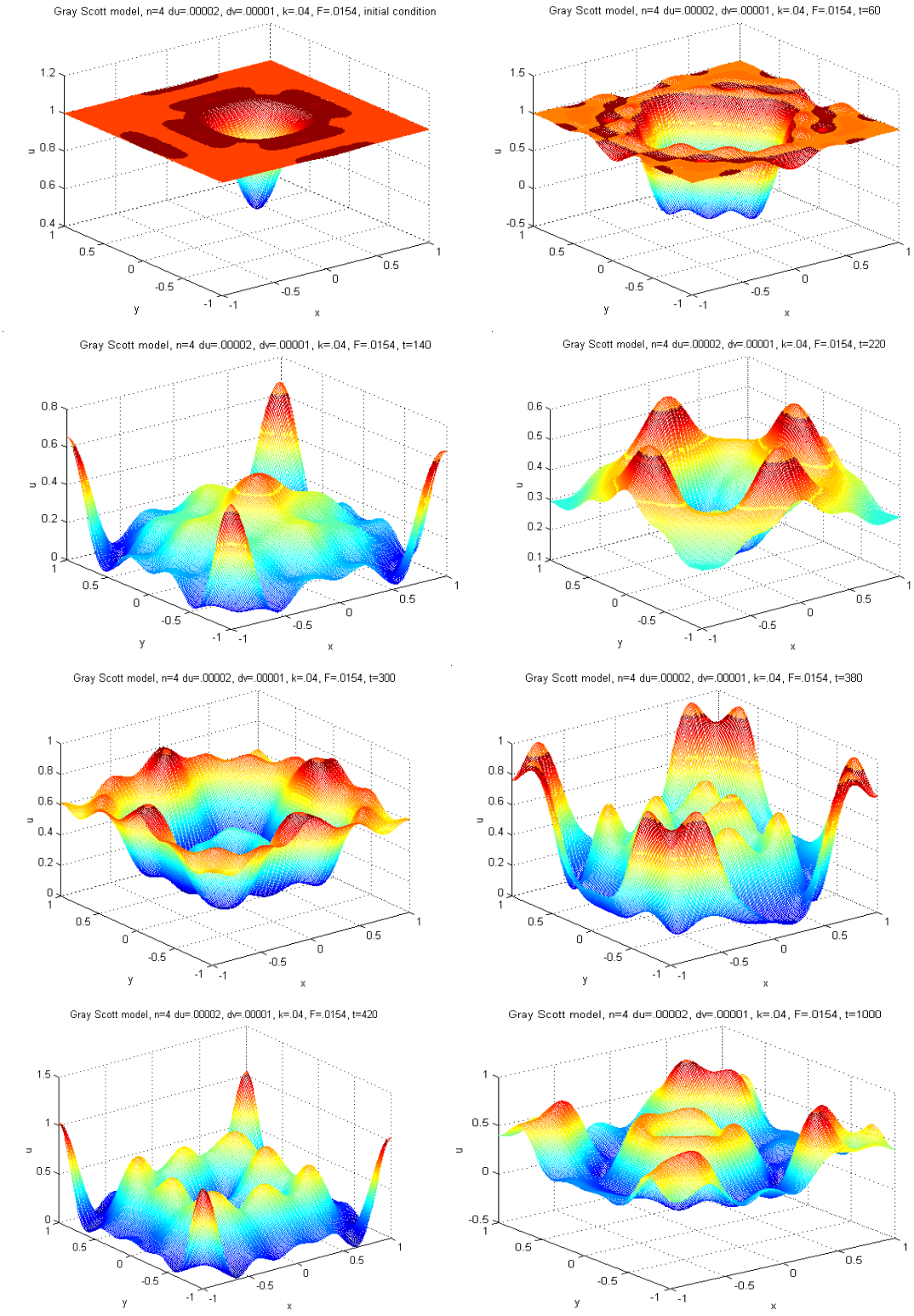
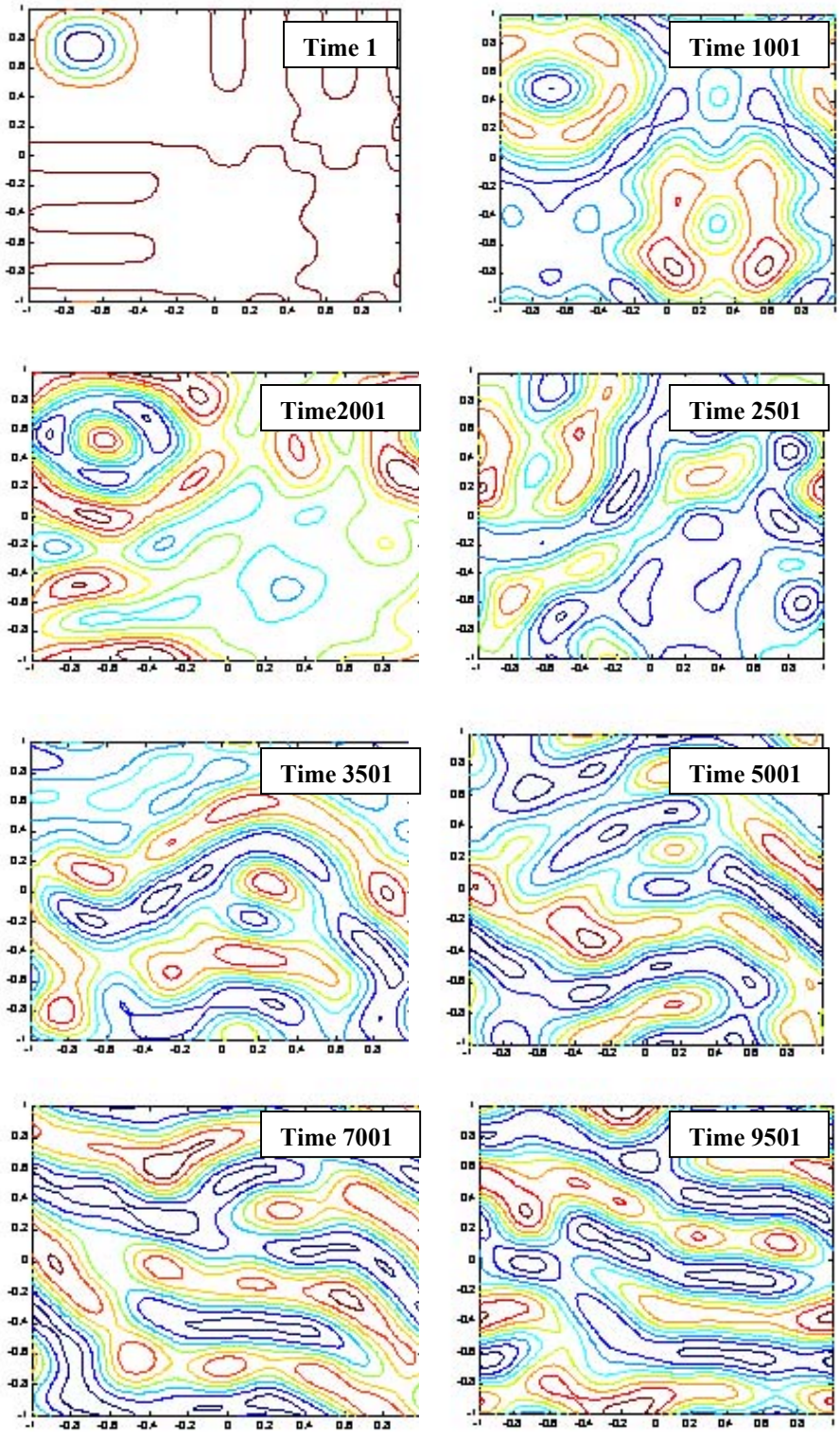
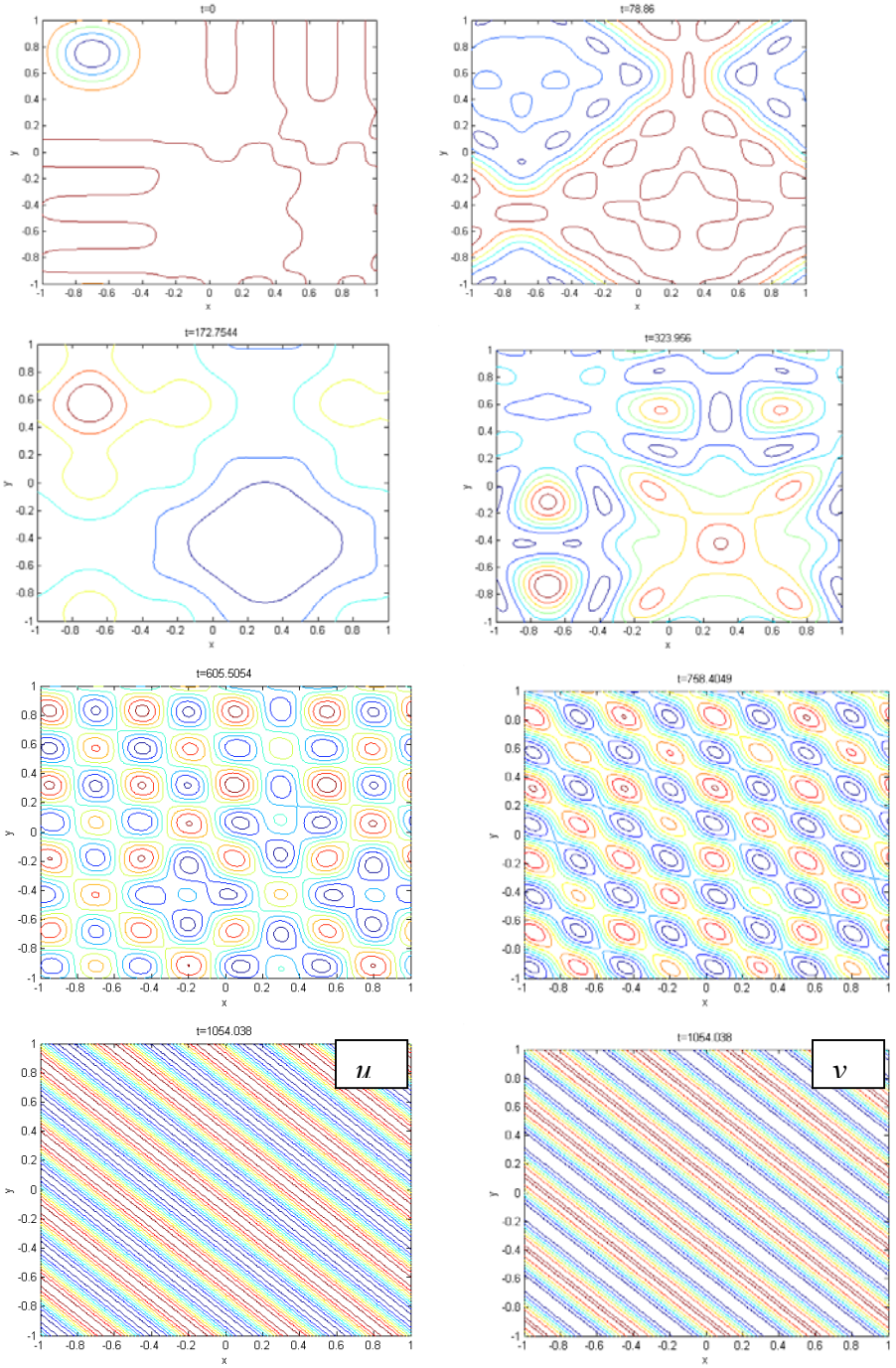


Figure 3.8 *u* for 4 basis functions at various times, *F*=0.0154, *k*=0.04



All of the patterns above have been periodic, oscillating with time in a predictable manner. Motivated by our analysis of diffusion driven instability, we alter the diffusion

Figure 5.5- Gray-Scott Model with new diffusion rates, d_u=0.0001, d_v=0.00001, N=4, F=0.0154, k=0.04, time up to 1000. The contour maps all represent the concentration of u where red is most concentrated and blue is least concentrated except for the final picture which shows the concentrations of v near time 1000 right beside its corresponding concentration u.



constants to obtain what seems to be a nonhomogeneous steady state solution. Figure 5.5 shows our simulated solution for N=4, F=0.0154, and k=0.04. This time our diffusion constants are $d_u = 10^{-4}$ and $d_v = 10^{-5}$. Now the diffusion constants differ by a multiple of 10 rather than 2 as they did in Figures 5.1-5.4. We start with a perturbation in the center this time. The initial condition on v is small positive perturbation from zero centered at the origin as well. As the simulation proceeds, we see that the concentration of u initially falls off from around the perturbation. Near the center, u forms a region of high concentration and then seems to break into four regions of increased concentration that spread towards the boundaries. This process of forming a concentrated region near the center and then breaking off repeats itself until the solution begins to settle into a checkerboard pattern of alternating high and low concentrations which then become stationary waves. Notice that the concentration of morphogens u and v are nearly complimentary at the final time. Gierer and Meinhardt showed in their activator/inhibitor reaction-diffusion system that the diffusion constants must differ significantly in order for patterns to occur [10].

We have shown numerical evidence to support the notion that reaction-diffusion equations have periodic solutions and nonhomogeneous steady state solutions. We have only used a few modes, and yet we have generated some complicated patterns and structures for the Gray-Scott model. Other researchers have been able to generate and classify many of the patterns that can be generated by various reaction-diffusion systems, ([5], [9],[11],[12],[14],[15],[17],[18],[19],[20],[24],[26],[28]). These systems, which are derived from the most basic physical and chemical laws, seem to take on a life of their

own given the right conditions. As research in this area proceeds, it will be interesting to see how well we can predict and control these self-organizing systems.

Bibliography

[1] Fletcher, C.A.J. *Computational Galerkin Methods*. Springer Series in Computational Physics. Springer-Verlag. New York 1984.

[2] Grindrod, Peter. *The Theory and Applications of Reaction-Diffusion Equations: Patterns and Waves*. 2nd Ed.:Oxford, Clarendon Press;1996

[3] Gollub, Gene H. and James M. Ortega. *Scientific Computing and Differential Equations: An Introduction to Numerical Methods.* Academic Press, INC. San Diego 1992.

[4] Gottlieb, David, and Steven Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications*. CBMS-NSF regional Conference series in applied mathematics. Society for Industrial and Applied Mathematics. Philadelphia 1997

[5] Gray P. and S.K Scott, Chem. Eng. Sci 38,29 1983; ibid 39 1087 (1984); J. Phys. Chem. 89, 22, (1985).

[6] Haberman, Richard, *Elementary Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems.* 3rd Ed. Prentice-Hall, Inc. Upper Sadle River, NJ 1998.

[7] Hoffman, K and R. Kunze, *Linear Algebra.* 2nd Ed. Prentice-Hall, Inc. Englewood Cliffs, NJ 1971.

[8] Kopell, Nancy. *Networks of neurons as dynamical systems: From Geometry to Biophysics.* Quarterly of Applied Mathematics **55** no.4, 707-718 (1998)

[9] Koch, A.J. and H. Meinhardt. "Biological Pattern Formation: from basic mechanisms to complex structures. *Rev. Mod. Phys.* 66 no. 4, 1481-1507 (Oct 1994)

[10] Kreyszig, Erwin. *Advanced Engineering Mathematics, 2nd Ed.* : John Wiley & Sons; 1993.

[11] Maini, Philip K. and Hans Othmer Eds. *Mathematical models for biological pattern formation.* Dedicated to Professor James D. Murray. The IMA Volumes in Mathematics and its Applications, 121. Frontiers in Application of Mathematics. Springer-Verlag, New York, 2001.

[12] Maini, Philip K. Maini, Gerhard C. Cruywagen, and James D. Murray. *Biological pattern formation on two-dimensional spatial domains: a nonlinear bifurcation analysis.* SIAM Journal of Applied Mathematics **57** no.6 1485-1509 (1997)

[13] Malek-Madani, Reza. *Advanced Engineering Mathematics: with Mathematica and Matlab*, Vol 2.:Addison Wesley Longman;1998

[14] Meinhardt, Hans. "Beyond Spots and Stripes: Generation of More Complex Patterns By Modifications and Additions of the Basic Reaction". In [8]. 143+

[15] Murray, John D. *Mathematical Biology.* Second ed. Springer-Verlag, Berlin 1989.

[16] Orszag, Steven A. "Spectral Methods for Problems in Complex Geometries". *Journal of Computational Physics.* 37, 70-92 (1980)

[17] Ouyang, Qi, and Harry L. Swinney. "Transition from a uniform state to hexagonal and striped Turing patterns." *Nature.* Vol. 352 610-612 (aug 1997)

[18] Pearson, John E. "Complex Patterns in a Simple System" *Science*. 261 189-194 (Jul 1993)

[19] Petrov, Valery, Qi Ouyang & Harry L. Swinney. "Resonant Pattern Formation in a Chemical System" *Nature*. Vol. 388 655-657, (aug 1997)

[20] Schofield, Peter, Mark Chaplain and Stephen Hubbard. "Mathematical Modelling of Host –Parisitoid Systems: Effects of Chemically Mediated Parasitoid Foraging Strategies on Within- and Between-generation Spatio-temporal Dynamics". *J. theor. Biol.* (2002) **212,** 31-47.

[21] Selkov, E.E. Eur. J. Biochem. 4 79 (1968))

[22] Sheisser, W.E. *The Numerical Method of Lines -- Integration of Partial Differential Equations*, San Diego: Academic Press Inc., 1991.

[23] Trefethen, Lloyd N. *Spectral Methods in MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematics. 2000

[24] Tsai, Leo L., Hutchison, Geoffrey R., and Enrique Peacock-Lopez. "Turing patterns in a self-replicating mechanism with a self complementary template." *Journal of Chem. Phys.* 113 no 5, 2003-2006.

[25] Turing, Alan M *The Chemical Basis of Morphogenesis*. Philisophical Transactions of the Royal Society of London **237** Aug (1952) . 37-72

[26] Williams, Roy *Xmorphia* http://www.cacr.caltech.edu/ismap/image.html cited 28 Feb 2001.

[27] Wolpert, Lewis. *The Triumph of the Embryo.* Oxford University Press, 1991,

[28] Wollkind, David J. and Laura E. Stephenson. "Chemical Turing Patterns: A Model System of a Paradigm for Morphogenesis". In [8] 113+

APPENDIX: MATLAB programs

Table Of Contents

Galerkin Method

BASIS

```
function z=basis(n,x,bas)
%this function sets up a basis function for the cosine basis, the
chebyshev basis, or the sin basis,
%the arguments are n- the integer corresponding to the basis function n
is any integer from zero up.
%x is the independant variable, in symbolic or numeric form.
%basis is one of three arguments:
%1 gives the basis cos(n*pi*x)
%2 gives the basis sin(n*pi*x)
%3 gives the chebyshev polynomial basis T(n)

if bas==1
   %cos basis
   z=cos(n*pi*x);
elseif bas==2
      %sin basis
      z=sin(n*pi*x);
elseif bas==3
      %chebyshev polynomial basis
      %it works symbolically, or numerically.  basis

      %symbolic clause
      if isa(x,'sym')==1
      n=n+1;
         %T(0)
      if n==1
            z=1;
                  %T(1)
            elseif n==2
            z=[1,0];
               %T(n) n>=3
            else
            %first it does it with the vectors and then converts
            %to syms
            %polynomials represented as vectors
            zn1=[1];
               z=[1,0];
            for j=3:n
                  zn2=zn1;
                  zn1=z;
                  z=conv([2,0],zn1);
                  z=z-[zeros(1,length(z)-length(zn2)),zn2];
            end

            %now converting the polynomial back to syms
         m=length(z);
            for k=m-1:-1:0
                  zx(m-k)=[x^(k)];
               end
            z=zx.*z;
```

```
            %this last line converts a vector of syms to the actual
polynomial
            z=sum(z);
            end
         return


      else
            %this loop does the chebyshev polynomial at a point x
            n=n+1;
            if n==1
            z=1;
            elseif n==2
            z=x;
            else
            zz(1,:)=ones(size(x));
            zz(2,:)=x;
               for j=3:n
               zz(j,:)=2*x.*zz(j-1,:)-zz(j-2,:);
            end
            z=zz(n,:);
            end
      end

else
   'Error-not a valid basis number.  Try help on BASIS to see what
numbers are valid'
   return
end
```

GAL_ALLENCAHNQUICK

```
function [u,x,t]=gal_allencahnquick(nn,eps2,tvec,xvec)
%[u,x,t]=gal_allencahnquick(4,.01,100,[-1:.01:1]);
%gal_pdedisplay(u,x,t);
%does the allen cahn equation ut=uxx+u-u^3 on -1..1
%with boundary conditions u(-1,t)=-1, u(1,t)=1,
%and initial condition u(x,0)=.53*x+.47*sin(-1.5*pi*x)

%setting the global variables for the nonlinearities and nonhomogeneous parts
for use in the deqn
global nonhomogeneous
global nonlinear1
global nonlinear2
global nonlinear3
global eps
eps=eps2;

tic,
aa=min(xvec);
bb=max(xvec);
for j=1:nn
   nonhomogeneous(j)=quadl(@nonhomog,aa,bb,[],[],j);
   initialcond(j)=quadl(@init,aa,bb,[],[],j);
   for k=1:nn
    nonlinear1(j,k)=quadl(@nonlin1,aa,bb,[],[],j,k);
       for m=1:nn
        nonlinear2(j,k,m)=quadl(@nonlin2,aa,bb,[],[],j,k,m);
```

```
            for q=1:nn
            nonlinear3(j,k,m,q)=quadl(@nonlin3,aa,bb,[],[],j,k,m,q);
            end
      end
   end
end,
toc
'global variables done'
tic,
[t,a]=ode45('gal_allencahnquickdeqn',tvec,initialcond);
'odes solved',
toc

tic,
x=xvec;
m=length(t);
for j=1:nn
   sinvec(j,:)=sin(j*pi*x);
end
size(sinvec)
size(x)
size(a)
size(a(1,:))
for j=1:m
   u(j,:)=x+a(j,:)*sinvec;
end,
toc


%functions for the global variables for the nonlinearities and nonhomogenous
parts
function y=nonhomog(x,j)
y=(x-x.^3).*sin(j*pi*x);

function y=init(x,j)
%y=(.53*x+.47*sin(-1.5*x*pi)-x).*sin(j*pi*x);
y=(.1*x+.9*sin(-1.5*x*pi)-x).*sin(j*pi*x);
function y=nonlin1(x,j,k)
y=x.^2.*sin(j*pi*x).*sin(k*pi*x);

function y=nonlin2(x,j,k,m)
y=x.*sin(j*pi*x).*sin(k*pi*x).*sin(m*pi*x);

function y=nonlin3(x,j,k,m,q)
y=sin(j*pi*x).*sin(k*pi*x).*sin(m*pi*x).*sin(q*pi*x);
```

GAL_ALLENCAHNQUICKDEQN

```
function dadt=gal_allencahnquickdeqn(t,a)
%differential equation called in gal_allencahnquick
global nonhomogeneous
global nonlinear1
global nonlinear2
global nonlinear3
global eps

nn=length(nonhomogeneous);
for j=1:nn
```

```
   nonlin2vec(j)=0;
   nonlin3vec(j)=0;
    for k=1:nn
        for m=1:nn
          nonlin2vec(j)=nonlin2vec(j)+a(k)*a(m)*nonlinear2(j,k,m);
          for q=1:nn
          nonlin3vec(j)=nonlin3vec(j)+a(k)*a(m)*a(q)*nonlinear3(j,k,m,q
          );
          end
        end
     end
  end
dadt=a-eps*pi^2*((1:nn)'.^2).*a+nonhomogeneous'-3*nonlinear1*a-
3*nonlin2vec'-nonlin3vec';
```

GAL_BURGERQUICK

```
function [t,x,u]=gal_burgerquick(NN,eps,tvec,xvec)
% this function finds a galerkin approximation to burgers' equation
% ut=eps*uxx-u*ux
%for dirichlet boundary conditions u(0,t)=u(1,t)=0 for x=[0,1]
%eps is the inverse of the reynold's number.
%sin(j*pi*x) is the basis function used
%the ODE solver used is ODE45
%
%inputs:  NN-- Number of basis functions used
%         eps-- diffusion parameter, inverse of reynold's number
%         tvec-- time of simulation- can be either a final time or a
vector of times
%               at which the solution is evaluated
%
%
%outputs:  t-- times at which solution is simulated, equal to tvec or
else output of ode45
%          x-- points along x axis where x is evaluated, equal to xvec
%          u-- output, values of u at points along x and t. u has
length(t) rows and length(x) columns
%
%Syntax:
%   [t,x,u]=gal_burgerquick(32,.01,1,[0:.01:1])
%
%See also: GAL_BURGERQUICKDEQN,GAL_BURGERDISPLAY, GAL_BURGER,
GAL_BURGERDEQN

tic,
%find initial values of a's
initfcn(1,2)
for j=1:NN
    init(j)=quadl(@initfcn,1/4,1/2,[],[],j);
end
'initial condition finished',
toc

%create P matrix where
P(i,j,m)=(sin(j*pi*x)*sin(k*pi*x)*pi*cos(m*pi*x))
%uses a trick
```

```
global P2 eps2
tic,
for j=1:NN
      for k=1:NN
         for m=1:NN
            if j+m==k
             P(j,k,m)=-pi*k/4;
         elseif abs(j-m)==k
             P(j,k,m)=pi*k/4;
              end
          end
      end
  end
'P matrix finished',
P2=P;
eps2=eps
toc

tic,
%do ODE routine
[t,a]=ode45('gal_burgerquickdeqn',tvec,init);
'odes solved',
toc

tic,
%reconstruct solution
x=xvec;
for j=1:length(t)
    u(j,:)=zeros(size(x));
   for k=1:NN
        u(j,:)=u(j,:)+a(j,k)*sin(k*pi*x);
    end
end,
'solution constructed',
toc

function z=initfcn(y,m)
z=sin(m*pi*y).*(1-cos(8*pi*y));
```

GAL_BURGERQUICKDEQN

```
function da=gal_burgerquickdeqn(t,a,eps)
%differential equation referenced in GAL_BURGERDEQN

global P2 eps2
NN=length(a);
for j=1:NN
    nonlin(j)=a'*P2(:,:,j)*a;
end
da=-pi^2*eps2*([1:NN]'.^2).*a-2*nonlin';
```

GAL_BURGERDISPLAY

```
function M=gal_burgerdisplay(t,x,u)
```

```
%utility for plotting the output of burger's equation files.
%
%inputs:  t-- times at which solution is simulated, equal to tvec or
else output of ode45
%          x-- points along x axis where x is evaluated, equal to xvec
%          u-- output, values of u at points along x and t. u has
length(t) rows and length(x) columns
%
%Output:  M- a movie that shows the evolution of burger's equation over
time
%         figure(1) contour plots of u(x,t)
%         figure(2) surface plot of u(x,t), to show changes over time
%
%see also:  GAL_BURGER, GAL_BURGERDEQN, GAL_BURGERQUICK,
GAL_BURGERQUICKDEQN

cla
%routine for the titles
blab=input('what basis? \n','s');
tspan=input('what time span? \n','s');
xspan=input('what domain? \n','s');
n=input('how many basis functions? \n')
%contour plots of u(x,t) v.s. x
figure(1)
view (2)
plot(x,u','k')
xlabel('x')
ylabel('u(x,t)')
title('Burger''s equation contour plot')
legend(['n=',int2str(n)],['basis=',blab],['time=',tspan],['x=',xspan])

%surface plot of u(x,t)in space and time
figure(2)
surf(x,t',u)
xlabel('X')
ylabel('time')
zlabel('u(x,t)')
title('Burger''s equation surface plot')
legend(['n=',int2str(n)],['basis=',blab],['time=',tspan],['x=',xspan])

j=input('movie? 1 if no, 2 if yes')
if j==1
    return
end

%movie routine
figure(3)
hold off
plot(x,u(1,:))
v=axis
xlabel('x')
ylabel('u(x,t)')
title('Burger''s equation movie')
m=length(t);
M=moviein(m);
```

```
for j=1:m
    plot(x,u(j,:))
    axis(v)
    M(:,j)=getframe;
    hold off
end
```

GAL_HEAT

```
%This script file finds a solution for the Heat Equation in one
dimension using the Galerkin Method
%on the interval [0,1]
%and then plots the solution at various times.
%the amount of time is an input t
%the number and type of basis functions are inputs n, bn.
%the boundary conditions u(t,0) and u(t,1) are inputs ut0,ut1.

clear all
%n is the number of basis functions
n=4

%bn indicates the type of basis function
%1 gives the basis cos(n*pi*x)
%2 gives the basis sin(n*pi*x)
%3 gives the chebyshev polynomial basis T(n)
bn=2

%these are the boundary conditions u(t,0)=ut0 and u(t,1)=ut1
ut0=1
ut1=2

%%degree of PDE for t
tdeg=1

%t is the time we let the simulation run
t=10

%sets up a as a matrix of the functions in time and their derivatives
%also sets up bv as a vector of symbolic basis functions
x=sym('x');
for j=0:n
  for k=0:tdeg
     a(k+1,j+1)=sym(['a',int2str(j),int2str(k)]);
  end
  bv(j+1)=basis(j,x,bn);
end
a
bv

%gives a generic linear function to satisfy the boundary conditions
u(0,t), u(1,t).  We will call this
%the inital condition function g=u(x,0)
g=(ut1-ut0)*x+ut0
```

```
%plugs the assumed solution into the left (eqnl) and right (eqnr) sides
of the equation.
%k is the heat constant
%f is the source (can be in the symbolic variable x if desired)

k=3
f=1
ya=a(1,:).*bv;
ya=[g,ya]
dyadt=a(2,:).*bv

eqnr=sum(k*diff(ya,x,2))+f
eqnl=sum(dyadt)


%multiplies both the left and right hand sides by the basis function
and then integrates from 0 to 1
for j=0:n
   vl(j+1)=int(eqnl.*bv(j+1),x,0,1);
   vr(j+1)=int(eqnr*bv(j+1),x,0,1);
end

global vl
global vr
vl
vr

%since
bc=zeros(n,1)
[t,avec]=ode45('gal_heatDEQN',10,bc);
xv=0:.01:1;

%avec(1,:).*bv
%sum(avec(1,:).*bv)
%subs(sum(avec(j,:).*bv),x,xv)
cla
hold on
for j=1:length(t)
    plot(xv,xv+1+subs(sum([0,avec(j,:)].*bv),x,xv))
end
%computes value for differential equation using the inital conditions
```

GAL_HEATDEQN

```
function s=gal_heatDEQN(t,v)
s(1)=2*(-1/(2*pi)*3*pi^3*v(1)+4/(2*pi));
s(2)=2*-6*pi^2*v(2);
s(3)=2*-1/(6*pi)*(81*pi^2*v(3)-4);
s(4)=2*-24*pi^2*v(4);
s=s';

%global vl
%global vr
```

```
%n=length(vl);

%dpolynorm=dpoly/dpoly(1);
%for j=1:n
 %  scoeff=sym2poly(vl(j));
  % vpoly=sym2poly(vr(j));
%  while length(vpoly<2);
%      vpoly=[0,vpoly];
%  end
%   vcoeff=vpoly(1);
%  vscalar=vpoly(2);
% s(j)=vcoeff/scoeff*v(j)+vscalar/scoeff;
%end
%s
```

GAL_PEARSON2D

```
function [u,v,ab,X,t,init]=gal_pearson2D(NN,F2,k2,Xvec,tvec)
%[u,v,ab,X,t,init]=gal_pearson2D(NN,F2,k2,Xvec,tvec)
%[u,v,ab,X,t,init]=gal_pearson2D(4,.0152,.04,[[-1:.01:1]',[-
1:.01:1]'],1000);
%this function solves Pearson's 2 morphogen reaction-diffusion equation
in 2 dimensions
%using the galerkin spectral method with periodic boundary conditions.
The resulting ODE's are solved
%using ODE45 (runge kutta 4 method, forward time differencing).
%
%the equations are defined as
%
%ut=du*laplacian(u)-uv^2+F(1-u)
%vt=dv*laplacian(v)+uv^2-(F+k)v
%
%u,v:  The concentrations of each morphogen
%F:  Dimensionless feed rate
%k:  Dimensionless rate constant
%du,dv:  Diffusion constants for u and v respectively
%
%%Ref:  Complex Patterns in a Simple System, John E. Pearson, Science,
V261, 5118, 9 Jul 1993 189-192
%
%Inputs:
%NN2=number of basis functions used
%F2,k2-scalar parameters
%Xvec- 2 column space vector, column one is the X coordinate, column 2
is the Y coordinate
%tvec- time vector discretizing time for ODE45
%
%Outputs:
%u,v- Morphogen concentrations
%X- Xvec
%t- tvec
%
%See GAL_PEARSON2DDEQN

%initialize global variables F,k,du,dv,parameters, NN, basis number,
M2, sparse
```

```
%inner product matrix
global F kk du dv
F=F2;
kk=k2;
du=.00002;
dv=.00001;


%finds the vector of constants for the Laplacian, G
global G
G=-NN:NN;
G=G.^2;
n=1;
for k=G;
    G2(:,n)=G'+k;
    n=n+1;
end
G=G2(:);

%finds initial condition
tic,
for p=-NN:NN
   for q=0:NN                    %-NN:NN
        %initu(p+NN+1,q+NN+1)=quadl(@initcondu,-
1,1,[],[],p)*quadl(@initcondu,-1,1,[],[],q);
        initu(p+NN+1,q+1)=dblquad(@initcondu,-1,1,-1,1,[],@quadl,p,q-
NN);
        initu(NN+1-p,2*NN+1-q)=conj(initu(p+NN+1,q+1));
        %initv(p+NN+1,q+NN+1)=quadl(@initcondv,-
1,1,[],[],p)*quadl(@initcondv,-1,1,[],[],q);
       initv(p+NN+1,q+1)=dblquad(@initcondv,-1,1,-1,1,[],@quadl,p,q-
NN);
      initv(NN+1-p,2*NN+1-q)=conj(initv(p+NN+1,q+1));
  end
end,
initu=initu(:)/4;
initv=initv(:)/4;
init=[initu;initv];
'init matrix finished',
toc

%Calling the ODE solver
%[t,ab]=ode45('gal_pearson2Dexpdeqn',tvec,init);,
tic,
[t,ab]=ode45('gal_pearson2Ddeqn',tvec,init);
'ode''s solved',
toc

%constructing the solution
tic,
x=Xvec(:,1)';
y=Xvec(:,2);
X=Xvec;
u=zeros(length(x),length(y),length(t));
v=zeros(length(x),length(y),length(t));
```

```
for j=1:length(t)
   for k=1:size(ab,2)/2
      p=mod(k,2*NN+1);
      if p==0
         p=2*NN+1;
      end
      p=p-(NN+1);
      q=ceil(k/(2*NN+1));
      q=q-(NN+1);
      modey=exp(i*pi*q*y);
      modex=exp(i*pi*p*x);
      u(:,:,j)=u(:,:,j)+ab(j,k)*modey*modex;
      v(:,:,j)=v(:,:,j)+ab(j,k+size(ab,2)/2)*modey*modex;
  end
end,
'solution constructed',
toc



function z=initcondu(x,y,p,q)    %,k)
%z=exp(-i*pi*k*x).*(1+.1*(x.^2-1));
%z=1-.5*(exp(-20*(x-.1).^2))*(exp(-20*(y)^2));    %-1/16*exp(-20*(x-
.1).^2)*exp(-30*(y-.1)^2);
z=1-.5*(exp(-20*(x+.7).^2))*(exp(-20*(y-.75)^2));    %-1/16*exp(-20*(x-
.1).^2)*exp(-30*(y-.1)^2);
z=exp(-i*pi*(p*x+q*y)).*z;

function z=initcondv(x,y,p,q)    %,k)
%z=exp(-i*pi*k*x).*(.1*(1-x.^2));
%z=.25*(exp(-20*(x-.05).^2))*(exp(-20*y^2));   %+1/16*exp(-20*(x-
.15).^2)*exp(-30*(y-.1)^2);
z=.25*(exp(-20*(x+.7).^2))*(exp(-20*(y-.65)^2));   %+1/16*exp(-20*(x-
.15).^2)*exp(-30*(y-.1)^2);
z=exp(-i*pi*(p.*x+q*y)).*z;
```

GAL_PEARSON2DDEQN

```
function dabdt=gal_pearson2Ddeqn(t,ab)
%two dimensional head equation differential equation called in
gal_heat2D.
%this program sets up the system of differential equations inolving the
a(m,n)(t)

global F kk G du dv
NN=(sqrt(length(ab)/2)-1)/2;

a=ab(1:length(ab)/2);
b=ab(length(ab)/2+1:end);
%does the a's without the nonlinear part.  The extra 4*Fv is added
because it is the (phi(0,0),F) term.  For every
%other equation, this disappears
```

```
Fv=zeros(size(a));
Fv(((2*NN+1)^2+1)/2)=F;


dabdtu=(-du*pi^2*G-F).*a+Fv;  %-sum(nonlinear);
%does the b's
dabdtv=(-dv*pi^2*G-F-kk).*b; %+sum(nonlinear);
for j=1:2*NN+1
    bm(:,j)=b((j-1)*(2*NN+1)+1:j*(2*NN+1));
end
bm2=[zeros(2*NN,6*NN+1);[zeros(2*NN+1,2*NN),bm,zeros(2*NN+1,2*NN)];zero
s(2*NN,6*NN+1)];

for q=-NN:NN
    for p=-NN:NN
        for k2=-NN:NN
            for k1=-NN:NN
                for m2=-NN:NN
                    for m1=-NN:NN

bmpq(k1+NN+1+(k2+NN)*(2*NN+1),m1+NN+1+(m2+NN)*(2*NN+1))=bm2((p-k1-
m1)+3*NN+1,(q-k2-m2)+3*NN+1);
                        %bmpq(k1k2,m1m2)
*bm2(p-k1-m1,q-k2-m2)
                    end
                end
            end
        end
        nonlin(p+NN+1+(q+NN)*(2*NN+1))=sum(sum((a*b.').*bmpq));
        %nonlin(p,q)                  =  sum(ak1,k2 x bm1*m2 x b(p-m1-
k1),(q-m2-k2)
    end
end
dabdtu=dabdtu-nonlin(:);
dabdtv=dabdtv+nonlin(:);
dabdt=[dabdtu;dabdtv];
```

GAL_1DPDEDISPLAY

```
function M=gal_1Dpdedisplay(t,x,u)
%type gal_1Dpdedisplay(t,x,u) to dislpay the results as a
%contour map, a surface plot or a movie
%plotting
cla

%routine for the titles
tit=input('input Title\n','s');
blab=input('what basis? \n','s');
tspan=input('what time span? \n','s');
xspan=input('what domain? \n','s');
n=input('how many basis functions')
%contour plots of u(x,t) v.s. x
figure(1)
view (2)
```

```
plot(x,u')
xlabel('x')
ylabel('u(x,t)')
title(tit)
legend(['n=',int2str(n)],['basis=',blab],['time=',tspan],['x=',xspan])

%surface plot of u(x,t)in space and time
figure(2)
surf(x,t',u)
xlabel('X')
ylabel('time')
zlabel('u(x,t)')
title([tit,'surface plot'])
legend(['n=',int2str(n)],['basis=',blab],['time=',tspan],['x=',xspan])

j=input('movie? 1 if no, 2 if yes')
if j==1
    return
end

%movie routine
figure(3)
hold off
plot(x,u(1,:))
v=axis
xlabel('x')
ylabel('u(x,t)')
title([tit,' movie'])
m=length(t);
M=moviein(m);

for j=1:m
    plot(x,u(j,:))
    axis(v)
    M(:,j)=getframe;
    hold off
end
```

GAL_2DPDEDISPLAY

```
function M=gal_2Dpdedisplay(u,X,t)
%produces movie images for given 2D pde data in the form:
%t= time vector
%X= space, x in column 1, y in column 2
%u= u(x,y,t)= the value of the function at x,y and t

cla
tit=input('Input Title\n','s')
%movie routine
f=input('Surface(2) or Contour (1)\n')
```

```
if f==1

hold off
x=X(:,1);
y=X(:,2);
xlabel('x')
ylabel('y')
title(tit)
m=length(t);
M=moviein(m);
for j=1:m
    contour(x,y,u(:,:,j))
    title(tit)
    M(:,j)=getframe;
    hold off
end


elseif f==2

hold off
x=X(:,1);
y=X(:,2);
u=real(u);
v=[-1,1,-1,1,min(min(min(u))),max(max(max(u)))];
xlabel('x')
ylabel('y')
zlabel('u(x,y,t)')
title(tit)
m=length(t);
M=moviein(m);
for j=1:m
    mesh(x,y,u(:,:,j))
    axis(v)
    title(tit)
    M(:,j)=getframe;
    hold off
end
else

   'wrong input'

end
```

GAL_ABRECONSTRUCT

```
function [u,v]=gal_abreconstruct(ab,t,init)
%this function reconstructs u and v from given ab vector, t, and init
values from
%2D pde results:
%
%inputs:  ab- length(t)xlength(init) vector of coefficient functions
%          t- times at which coefficient functions are evaluated
%          init-  initial conditions for the ab same as ab(:,1)
%
```

```
%syntax:  [u,v]=gal_abreconstruct(ab,t,init)

tic,
x=[-1:.01:1];
y=x';

u=zeros(length(x),length(y),length(t));
v=zeros(length(x),length(y),length(t));
NN=(sqrt(length(ab)/2)-1)/2;


for j=1:length(t)
   for k=1:size(ab,2)/2
      p=mod(k,2*NN+1);
      if p==0
         p=2*NN+1;
      end
      p=p-(NN+1);
      q=ceil(k/(2*NN+1));
      q=q-(NN+1);

      modey=exp(i*pi*q*y);
      modex=exp(i*pi*p*x);
      u(:,:,j)=u(:,:,j)+ab(j,k)*modey*modex;
      v(:,:,j)=v(:,:,j)+ab(j,k+size(ab,2)/2)*modey*modex;
  end
end,
'solution constructed',
toc
```

Gray-Scott Analysis

GAL_2DGS_N0EIGENVALS

```
%eigenvalue movie for the diffusionless Grey Scott Model
%
%   a_t=-a*b^2-F(1-a)
%   b_t=a*b^2+(F+k)b
%
%This routine creates a movie of the eigenvalues of the derivative
matrix of {a_t(a,b),b_t(a,b)}
%evaluated at the equilibrium points of the system in the complex plane
when k is held constant
%at .04 and F varies in order to examine local stability of the
equilibrium points.

cla
hold on
k=.04
xlabel('real axis')
ylabel('imaginary axis')
n=1
Fi=.01
```

```
Ff=.17
df=.0001
for j=Fi:df:Ff
    F=j;
    eq0=[-F,0;0,-(F+k)];
    eig0=eig(eq0);
    figure(1)
    axis([-.25,.15,-.15,.15]);
    hold on
    plot(eig0,zeros(size(eig0)),'x')
    r=roots([F+k,-F,F^2+F*k]);
    r=sort(r);
    b=r(1);
    a=(F+k)/b;
    eq1=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)];
    eig1=eig(eq1);
    figure(2)
    axis([-.25,.15,-.15,.15]);
    hold on
    if isreal(b)==1
        if isreal(eig1)==1
            plot(eig1,zeros(size(eig1)),'x')
        else
            plot(eig1,'x')
        end
    end

    b=r(2);
    a=(F+k)/b;
    eq2=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)];
    eig2=eig(eq2);
    figure(3)
    axis([-.25,.15,-.15,.15]);
    hold on
    if isreal(b)==1
        if isreal(eig2)==1
            plot(eig2,zeros(size(eig2)),'x')
        else
            plot(eig2,'x')
        end
    end
    %M(:,n)=getframe;
    n=n+1;
end
figure(1)
plot([0,0],[-.15,.15],'k')
plot([-.25,.15],[0,0],'k')
xlabel('real axis')
ylabel('imaginary axis')
text(-.25,.15,'Eigenvalues of the stability matrix of (a_0,b_0)')
text(-.23,.125,['k=',num2str(k),', F increases from ',num2str(Fi),' to
',num2str(Ff)]);

figure(2)
plot([0,0],[-.15,.15],'k')
```

```
plot([-.25,.15],[0,0],'k')
xlabel('real axis')
ylabel('imaginary axis')
text(-.25,.15,'Eigenvalues of the stability matrix of (a_1,b_1)')
text(-.23,.125,['k=',num2str(k),', F increases from ',num2str(Fi),' to
',num2str(Ff)]);

figure(3)
plot([0,0],[-.15,.15],'k')
plot([-.25,.15],[0,0],'k')
xlabel('real axis')
ylabel('imaginary axis')
text(-.25,.15,'Eigenvalues of the stability matrix of (a_2,b_2)')
text(-.23,.125,['k=',num2str(k),', F increases from ',num2str(Fi),' to
',num2str(Ff)]);
```

GAL_2DGS_HOPFFIND

```
%find value of F where eig2 crosses the axis and becomes stable
k=.04-->F=.0152786
%answer gives estimate that is less than the true answer by no more
than tol
cla
hold on
n=1
tol=10^-10
for k=0:.0005:.0625
    %finds starting F value
    F=roots([-4,1-8*k,-4*k^2,0]);
    F=F(F>0);
    Fmin=min(F);
    Fmax=max(F);
    eq3bot(n,:)=[k,Fmin];
    eq3top(n,:)=[k,Fmax];
    %adds a bit to starting value to be sure that the equilibrium is
real but not
    %enough to make it stable
    F=Fmin+.00001;
    %initializes eig2 for while loop
    r=roots([F+k,-F,F^2+F*k]);
    r=sort(r);
    b=r(2);
    a=(F+k)/b;
    eq2=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)];
    eig2=eig(eq2);

    %set initial step value
    d=.001;
    %outer while loop decreases step value until it is within tolerance
    while d>tol
        %inner while loop tests eig2 for stability
        while real(eig2)>0
            F=F+d;
            r=roots([F+k,-F,F^2+F*k]);
            r=sort(r);
```

```
            b=r(2);
            a=(F+k)/b;
            eq2=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)];
            eig2=eig(eq2);
        end
        %sets F to just above where it broke the while loop so it can
go through again at
        %higher tolerance with a reinitialized eig(2)
        F=F-d;
        r=roots([F+k,-F,F^2+F*k]);
        r=sort(r);

        b=r(2);
        a=(F+k)/b;
        eq2=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)];
        eig2=eig(eq2);

        %decreases step value
        d=d*.1;
    end
    %stores answers
Fhopf(n)=F;
n=n+1;
end

eq3=[eq3bot([2:end],:);flipud(eq3top)];
plot(eq3(:,1),eq3(:,2))
kvec=eq3bot(2:end,1);
Fhopf=Fhopf(2:end);
plot(kvec',Fhopf,'--')
```

GAL_2DGS_FIELDPLOTS

```
%plots fieldlines and equilibrium points.  Can be altered to make a
movie as F is varied
n=1
figure(2)
for j=.0154
cla
hold on

F=j
k=.04
umin=0
umax=1
vmin=0
vmax=1
spa=20
time=100

%plotting the roots of the equation
plot(1,0,'ro','markersize',15)
eq0=[-F,0;0,-(F+k)]
```

```
eig0=eig(eq0)

r=roots([F+k,-F,F^2+F*k]);
r=sort(r);
if isreal(r)==1
    plot((F+k)./r(1),r(1),'r+','markersize',15)
    plot((F+k)./r(2),r(2),'ro','markersize',15)
end
text(umax-.1*(umax-umin),vmax-.1*(vmax-
vmin),[num2str(umax),',',num2str(vmax)]);
text(umin+.1*(umax-umin),vmin+.1*(vmax-
vmin),[num2str(umin),',',num2str(vmin)]);


b=r(1);
a=(F+k)/b;
eq1=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)]
eig1=eig(eq1)

b=r(2);
a=(F+k)/b;
eq2=[-b^2-F,-2*a*b;b^2,2*a*b-(F+k)]
eig2=eig(eq2)

for j=linspace(umin,umax,spa)
    for m=linspace(vmin,vmax,spa)
        [t,u]=ode45('gal_pearson0deqn',time,[j,m],[],F,k);
        plot(u(:,1),u(:,2))
    end
end

xlabel('u')
ylabel('v')
title(['F=',num2str(F),' k=',num2str(k), ' time=',num2str(time)])
axis([umin,umax,vmin,vmax])
%M(:,n)=getframe;
n=n+1;
end
```

GAL_PEARSON0DEQN

```
%function used in Gray Scott Analysis functions for the N=0
differential equation
function dabdt=gal_pearson0deqn(t,ab,flag,F,k)
dabdt(1,1)=-ab(1).*ab(2).^2+F*(1-ab(1));
dabdt(2,1)=ab(1).*ab(2).^2-(F+k)*ab(2);
```

GAL_DDINSTABILITY

```
clear all
F=.0154
k=.04
du=.00002
dv=.00001

n=1;
modes=0:10:5000;
for mn=modes;
    b=(F+sqrt(F^2-4*F*(F+k)^2))/(2*(F+k));
    a=(F+k)/b;
    M=[-b^2-F-mn*du,-2*a*b;b^2,2*a*b-(F+k)-mn*dv];
    eva=eig(M);
    y(n)=real(eva(2));
    n=n+1;
end
plot(modes,y);
```